

Final report of ITS Center project: Hampton Roads simulation

For the Center for ITS Implementation Research

A U.S. DOT University Transportation Center

“Development of ITS Evaluation Test-Bed Using Microscopic Simulation – City of Hampton Case Study”

Principal Investigator:

Byungkyu “Brian” Park

August 2003

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.



Research Report No. UVACTS-15-0-45
August 2003

Development of ITS Evaluation Test-Bed Using Microscopic Simulation – City of Hampton Case Study

By:

Ilsoo Yun

Byungkyu “Brian” Park

A Research Project Report

**For the Center for ITS Implementation Research (ITS)
A U.S. DOT University Transportation Center**

Ilsoo Yun

Email: iy6m@virginia.edu

Dr. Byungkyu “Brain” Park

Department of Civil Engineering

Email: bpark@virginia.edu

Center for Transportation Studies at the University of Virginia produces outstanding transportation professionals, innovative research results and provides important public service. The Center for Transportation Studies is committed to academic excellence, multi-disciplinary research and to developing state-of-the-art facilities. Through a partnership with the Virginia Department of Transportation's (VDOT) Research Council (VTRC), CTS faculty hold joint appointments, VTRC research scientists teach specialized courses, and graduate student work is supported through a Graduate Research Assistantship Program. CTS receives substantial financial support from two federal University Transportation Center Grants: the Mid-Atlantic Universities Transportation Center (MAUTC), and through the National ITS Implementation Research Center (ITS Center). Other related research activities of the faculty include funding through FHWA, NSF, US Department of Transportation, VDOT, other governmental agencies and private companies.

Disclaimer: The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

CTS Website

<http://cts.virginia.edu>

Center for Transportation Studies

University of Virginia

351 McCormick Road, P.O. Box 400742

Charlottesville, VA 22904-4742

434-924-6362

1. Report No. VACTS-15-0-45	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Development of ITS Evaluation Test-Bed Using Microscopic Simulation - City of Hampton Case Study		5. Report Date August, 2003
		6. Performing Organization Code
7. Author(s) Ilsoo Yun and Byungkyu "Brian" Park		8. Performing Organization Report No.
9. Performing Organization and Address Center for Transportation Studies University of Virginia PO Box 400742 Charlottesville, VA 22904-7472		10. Work Unit No. (TRAIS)
		11. Contract or Grant No.
12. Sponsoring Agencies' Name and Address Office of University Programs, Research and Special Programs Administration US Department of Transportation 400 Seventh Street, SW Washington DC 20590-0001		13. Type of Report and Period Covered Final Report
		14. Sponsoring Agency Code
15. Supplementary Notes		
16. Abstract Microscopic traffic simulation models are very powerful tools as they provide inexpensive, fast, and risk-free evaluation environment. They not only provide the simulation of scenarios that cannot be practically tested in real world conditions, but also allow various network wide performance measures including travel times, delay and emissions. In addition, traffic simulation models are also being used extensively in Intelligent Transportation Systems (ITS) researches. However, there is a need to develop appropriate methodology and gain experience in practical usage of traffic simulation models to evaluate ITS deployments. In this study, the procedure relevant to building a microscopic traffic simulation-based test-bed for ITS applications is presented. This thesis describes the entire process for building a microscopic traffic simulation-based test-bed for ITS using a case study. The process consists of building a basic traffic simulation network, the API development for coordinated actuated signal control and, the dynamic O-D matrix estimation for the network. In order to estimate the dynamic O-D matrix, an approach using GA and QUEENSOD method coupled with traffic simulation model is introduced. Some findings during making the simulation test-bed are also presented. The findings are very useful for developing a simulation-based test-bed because the lessons learned from this study can reduce trial-and-errors and efforts needed.		
17 Key Words Intelligent Transportation Systems (ITS), evaluation, microscopic traffic simulation	18. Distribution Statement No restrictions. This document is available to the public.	

ABSTRACT

Microscopic traffic simulation models are very powerful tools as they provide inexpensive, fast, and risk-free evaluation environment. They not only provide the simulation of scenarios that cannot be practically tested in real world conditions, but also allow various network wide performance measures including travel times, delay and emissions. In addition, traffic simulation models are also being used extensively in Intelligent Transportation Systems (ITS) researches. However, there is a need to develop appropriate methodology and gain experience in practical usage of traffic simulation models to evaluate ITS deployments. In this study, the procedure relevant to building a microscopic traffic simulation-based test-bed for ITS applications is presented.

This thesis describes the entire process for building a microscopic traffic simulation-based test-bed for ITS using a case study. The process consists of building a basic traffic simulation network, the API development for coordinated actuated signal control and, the dynamic O-D matrix estimation for the network. In order to estimate the dynamic O-D matrix, an approach using GA and traffic simulation model is introduced in Chapter 5. Some findings during making the simulation test-bed are presented in Chapter 3, 4 and 5. The findings are very useful for developing a simulation-based test-bed because the lessons learned from this study can reduce trial-and-errors and efforts needed.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
	1.1 BACKGROUND	1
	1.2 PROBLEM STATEMENT	2
	1.3 RESEARCH OBJECTIVES AND METHODOLOGY	4
	1.4 SCOPE OF STUDY	5
	1.5 ORGANIZATION OF THIS THESIS	5
2	LITERATURE REVIEW	6
	2.1 INTRODUCTION	6
	2.2 DYNAMIC O-D MATRIX ESTIMATION	6
	2.2.1 STATIC O-D ESTIMATION	6
	2.2.1 Static O-D Estimation	6
	2.2.2 Dynamic O-D Estimation	8
	2.2.3 Estimation of Temporal Route Choice for Dynamic O-D Estimation	16
	2.3 GENETIC ALGORITHM	17
	2.3.1 Introduction	18
	2.3.2 Mechanisms of Genetic Algorithm	18
	2.3.3 Genetic Algorithms in O-D Estimation	24
	2.4 TRAFFIC SIMULATION MODEL IN ITS EVALUATION	27
	2.5 SUMMARY	28
3	A DEVELOPMENT OF LARGE SCALE CASE STUDY SIMULATION NETWORK	29
	3.1 INTRODUCTION	29
	3.2 MICROSCOPIC SIMULATION MODEL SELECTION	30
	3.3 MICROSCOPIC SIMULATION MODEL, PARAMICS	33
	3.3.1 User Interface	33
	3.3.2 Network Structure	33
	3.3.3 Traffic Generation and Assignment	33
	3.3.4 Signal Control in PARAMICS	34
	3.3.5 API in PARAMICS	34
	3.3.6 Model Weaknesses	35
	3.4 PARAMICS NETWORK	35
	3.4.1 Network Building	35
	3.4.2 Signal Control Logics	39
	3.4.3 Parameter Setting	39
	3.4.4 Simulation	39
	3.4.5 Visualization	40

Chapter		Page
	3.5 DEVELOPMENT OF AN API FOR COORDINATED ACTUATED SIGNALS	42
	3.5.1 Geometry of Intersection and Detectors	42
	3.5.2 Yellow Change Interval	44
	3.5.3 New Phase Sequence for API	45
	3.5.4 Input Data Structure	47
	3.5.5 Control Logic	49
	3.5.6 Fully Actuated Signal Control and T-intersection	50
	3.6 SUMMARY	51
4	BUILDING DYNAMIC O-D ESTIMATION MODELS USING GA AND QUEENSOD MODEL	53
	4.1 INTRODUCTION	53
	4.2 USE OF ASSIGNMENT MATRIX	55
	4.3 QUEENSOD MODEL	57
	4.3.1 Mechanism of QUEENSOD Model	57
	4.3.2 Using Existing Traffic Information	59
	4.3.3 QUEENSOD-based O-D Estimation Process	60
	4.4 GA-BASED O-D ESTIMATION MODEL	61
	4.4.1 Determination of GA Selection Method, GA Parameters and GA Operators	62
	4.4.2 Selection of Solution Representation, GA model and Evaluation Function	65
	4.4.3 Evaluation Function	72
	4.4.4 GA-based O-D Estimation Process	77
	4.5 SUMMARY	79
5	IMPLEMENTATION OF DYNAMIC O-D MATRIX ESTIMATION MODELS	80
	5.1 INTRODUCTION	80
	5.1.1 Data Collection	80
	5.1.2 Peak Period Selection	81
	5.2 QUEENSOD-BASED DYNAMIC O-D ESTIMATION	82
	5.2.1 Finding Better Assignment Matrix (Step #1)	82
	5.2.2 Finding the Best O-D Matrix (Step #2)	84
	5.3 GA-BASED DYNAMIC O-D ESTIMATION	91
	5.3.1 Finding Better Assignment Matrix (Step #1)	92
	5.3.2 Finding Appropriate Total O-D Demand (Step #2)	94
	5.3.3 Finding Best Sum of O-D Demands and O-D Matrix (Step #3)	95
	5.4 SUMMARY	98

Chapter	Page
6 CONCLUSIONS AND RECOMMENDATIONS	100
6.1 CONCLUSIONS	100
6.2 RECOMMENDATIONS	102
REFERENCES	104
APPENDIX A PHASE SEQUENCE OF 25 ORDERS IN API	108
APPENDIX B INPUT FILES OF API FOR COORDINATED ACTUATED CONTROLLER	110
APPENDIX C SOLUTION DECODING CONCEPTS IN THE GA	116
APPENDIX D MATALAB CODES FOR QUEENSOD-BASED MODEL AND GA-BASED MODEL	119

TABLE OF FIGURES

Figure		Page
1	Example of Chromosome-like Data Structure Using Binary Digits	19
2	City of Hampton, Virginia	30
3	GIS Map Used as Reference for Network Coding	37
4	Images in Lizard Tech's MrSID Format	38
5	PARAMICS Network for the Case Study	38
6	Example of inappropriate "next lane"	41
7	Impact of Insufficient Distance for "signposting"	41
8	An Example Intersection in PARAMICS Network	43
9	Splits and Phasing Diagram Based on a Dual Ring, Eight-Phase Controller	45
10	Splits and Phasing Diagram Based on a Single Ring Controller in PARAMICS	45
11	Seven External Zones of Case Study Network	60
12	Selection Probability Comparison of Two GA Selection Methods	64
13	Three Stages of Making O-D Matrix Using Solution Representation 1	68
14	Three Stages of Making O-D Matrix Using Solution Representation 2	69
15	Three Stages of Making O-D Matrix Using Solution Representation 3	70
16	Estimated and Observed Link Flows Comparison	74
17	Correlation of Estimated Link Flows from GA Using SSE and MAPE	74
18	15-minute Interval Link Flows	81
19	Convergence of the QUEENSOD Model in Step 1	82
20	Change of Total O-D Demand in Iterations	83
21	Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS	84
22	Convergence of the QUEENSOD Model In Step 2	85
23	Change of Total O-D Demand in Iterations	85
24	Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS	88
25	O-D Demands Adjustment by the QUEENSOD Model	90
26	Comparison of Estimated Link Flows vs. Observed Link Flows	91
27	Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS	93
28	Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS	94
29	Convergence of GA Iterations	95
30	Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS	96
31	Comparison of Estimated Link Flows vs. Observed Link Flows	99

TABLE OF TABLES

Table		Page
1	Summary of Simulation Run Times Comparison	32
2	Proposed Phase Sequence for API in PARAMICS	46
3	Example of an Assignment Matrix	56
4	Summary of Initial Populations From Three Solution Representations	71
5	Summary of Examination	72
6	Error Measures in Divided Groups	76
7	Results of the QUEENSOD Iterations	86
8	Results of O-D Matrix Estimates Evaluation Using PARAMICS	87
9	Results of the GA Runs	96
10	Evaluation Results of Estimated O-D Matrix Using PARAMICS	97

LIST OF ACRONYMS AND ABBREVIATIONS

API - Application Programming Interface

DTA – Dynamic Traffic Assignment

GA – Genetic Algorithm

GLS – Generalized Least Squares

ITS – Intelligent Transportation Systems

LRE – Least Relative Error

LSE – Least Square Error

MAPE – Mean Absolute Percent Error

MLE – Maximum Likelihood Estimation

MOE - Measures of Effectiveness

O-D – Origin-Destination

SSE – Sum of Squared Error

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Microscopic traffic simulation models are very powerful tools as they provide inexpensive, fast, and risk-free evaluation environment. They have been widely used for evaluating traffic operation and traffic management studies. They not only allow to run simulation scenarios that may not be practically tested in a real world condition, but also provide various network wide performance measures including travel times, emissions, etc (1). It appears that microscopic traffic simulation is slowly becoming an accepted tool in the traditional transportation engineering applications, such as signal timing plan development and capacity studies (2, 3). In addition, traffic simulation models are also being used extensively in the Intelligent Transportation Systems (ITS) researches. When a microscopic traffic simulation model is used to evaluate ITS projects, it is necessary to ensure that the model reflects real world conditions. Hence high fidelity stochastic and

microscopic simulation models are used and the models are calibrated using field traffic data.

However, there is a need to develop appropriate methodology for building the simulation models and to gain experience in practical usage of traffic simulation models to evaluate various ITS deployments. In this research, the procedure relevant to developing a microscopic traffic simulation-based ITS evaluation test-bed is presented. The procedure includes building a traffic simulation network and estimating a dynamic Origin-Destination (O-D) matrix.

1.2 PROBLEM STATEMENT

As transportation system networks to be analyzed become more and more complex, tools that are much better than traditional planning models are required to perform such analyses. Microscopic traffic simulation models used for traffic studies can realistically replicate the movement of traffic to match existing observed conditions and produce network-level performance measures. Thus, traffic simulation models offer several advantages over traditional planning models. The benefits include more detailed results, fancier graphics, capability to model ITS applications, and over saturated facilities. This is why microscopic simulation models have been widely used as test-beds with increasing attention to ITS (1).

However, users should be careful with the development of the network. It is necessary for the network to replicate real word situations to the highest degree possible in order to establish the functionality of simulation model. In particular, when the

network is big, the user might receive abnormal results during simulation because a large-scale simulation network causes more complex dynamics in interactions of vehicle-to-vehicle and vehicle-to-traffic control as well as efforts for data surveying and entering. This thesis builds a large network and presents the efforts to make the traffic simulation network to be more realistic.

Another issue for traffic simulation models when they are used in large networks is gathering an appropriate Origin-Destination (O-D) trip matrix. Conventional methods for collecting O-D matrix tend to be costly, labor intensive and time consuming. Therefore, the estimation of O-D matrices from traffic counts is of great interest as traffic counts are easily available, inexpensive to collect and they do not disrupt traffic (4). Moreover, obtaining more accurate and complete traffic information can be expected through good surveillance systems such as vehicle detection sensors, automatic vehicle location (AVL) systems, automatic vehicle identification (AVI) systems, and toll collection systems (TCS). Because of these reasons the process of O-D estimation from traffic counts is gaining interests.

For several decades, many researchers have proposed and introduced various techniques to estimate static O-D demands. In conventional planning applications, the static O-D matrix is very valuable. However, applications based on static O-D matrices can not explain several issues like the dynamics of congestions, departure time selection, and time-varying link usage and its spatial change over time. These limitations become more severe when the estimated O-D matrix is used for short-term planning studies or traffic operational studies. And with the introduction of ITS in traditional transportation

domain, there has been more need for the dynamic O-D matrices to consider the temporal and spatial change in congestion and behaviors of drivers.

Recent researches in the dynamic O-D matrix estimation based on observed traffic counts provided promising results on simple networks, long freeway sections and complex corridors. However, only a few researchers tried to apply the dynamic O-D matrix estimation techniques to a large city. This thesis proposes a new approach for the dynamic O-D matrix estimation in a large-scale network.

1.3 RESEARCH OBJECTIVES AND METHODOLOGY

The objectives of this thesis are: 1) to build a large-scale traffic simulation network using PARAMICS, 2) to develop an coordinated actuated signal control logics for the network, and 3) to estimate a dynamic O-D matrix for the network. The specific tasks are as follows:

1. Review relevant literature and assess the state-of-the-art in the dynamic O-D estimation;
2. Build a traffic simulation test-bed (base network) using PARAMICS;
3. Develop coordinated actuated signal control logic for the test-bed; and
4. Estimate a dynamic O-D matrix using QUEENSOD and Genetic Algorithm (GA).

1.4 SCOPE OF STUDY

The traffic simulation test-bed consists of the traffic simulation network for the entire City of Hampton and estimates of the dynamic O-D matrix for the network. As a microscopic traffic simulation model, PARAMICS was selected.

1.5 ORGANIZATION OF THIS REPORT

This report is organized into six chapters. Chapter 1 is an introductory chapter. Chapter 2 reviews the state-of-the-art related to the dynamic O-D matrix estimations. It includes the introduction of various techniques for the dynamic O-D matrix estimation and their applications and also discusses genetic algorithm (GA) and its applications in the O-D matrix estimation. The developments of the basic simulation network for the case study and the API for a coordinated actuated signal controller are explained in Chapter 3. Chapter 4 and 5 detail the process of estimating a dynamic O-D demand matrix for traffic simulation model based on counted link flows. Finally, Chapter 6 describes the conclusions and recommendations of this project.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter performs a review of literature relevant to dynamic O-D matrix estimations, a genetic algorithm (GA) and use of microscopic traffic simulation model in ITS evaluation. Dynamic O-D matrix estimation methods are summarized and then, the introduction of GA and its applications on the static O-D matrix estimation are followed.

2.2 DYNAMIC O-D MATRIX ESTIMATION

2.2.1 Static O-D Estimation

An O-D matrix estimation by using traffic information can be static or dynamic (i.e., time independent or time dependent). The both static and dynamic estimations generally calculate O-D matrices based on the relationship between traffic information (i.e., traffic counts) and O-D information. Ortuza and Willumsen (4) mathematically expressed the relationship as follow:

$$v_a = \sum_{ij} T_{ij} p_{ij}^a, \quad 0 \leq p_{ij}^a \leq 1 \quad (1)$$

where,

- v_a = observed traffic count in link a ,
- T_{ij} = trips generated from zone i to zone j ,
- p_{ij}^a = proportion of trips from zone i to zone j traveling through link a ,
- i = origin,
- j = destination, and
- a = link identifier.

The assignment parameter (p_{ij}^a) is used to define the proportion of trips from zone i to zone j traveling through link a . Thus, the observed traffic count (v_a) in a particular link a is the summation of the contributions of all trips between zones to that link. The p_{ij}^a can be obtained using various trip assignment techniques ranging from a simple all-or-nothing to a more complicated equilibrium assignment. Using the calculated p_{ij}^a and the observed traffic counts (v_a), the unknown T_{ij} is estimated.

For several decades researchers have proposed various models and their solution techniques to estimate static O-D matrices. Based on theory used in the OD estimation, the models can be divided into gravity-based models, entropy models, equilibrium models, statistical models, neural network models, fuzzy weight models (5), and GA models (6, 7). The gravity-based models utilize multiple linear regression or non-linear regression (4, 5). Some equilibrium models use a modified Frank-Wolf algorithm (5). The maximum entropy models generally have their own specific computer programs (4).

The statistical models utilize the generalized least square estimation methods, Bayesian inference methods, maximum likelihood estimation or bilevel programming method (8).

In conventional planning applications, the static O-D matrix is sufficient. However, the applications based on these static O-D matrices do not capture departure time selections, dynamics of congestions, and time-varying link usage and its spatial change over time. With the introduction of ITS in traditional transportation area, there has been an increasing need for the dynamic O-D matrix to consider the temporal and spatial change in congestion and drivers' behaviors.

2.2.2 Dynamic O-D Estimation

Dynamic O-D matrix estimation involves finding not only trips from zone i to zone j but also the departure times of the trips. In other words, dynamic O-D matrix estimation searches the trips (T_{ij}^{dt}) leaving zone i to zone j during time interval dt so as to match the estimated traffic counts to surveyed traffic counts using assignment parameters ($p_{ij,dt}^{a,t}$).

The variable $p_{ij,dt}^{a,t}$ is used to define the proportion of trips leaving from zone i to zone j during time interval dt and traveling through link a during time interval t . Thus, the observed traffic count (v_a^t) in a particular link a during time interval t is the summation of the temporal and spatial contributions of all trips between zones to that link.

Mathematically, the relationship between traffic information (i.e., traffic counts) and O-D information can be expanded from Equation (1) as follow:

$$v_a^t = \sum_{dt} \sum_{ij} T_{ij}^{dt} p_{ij,dt}^{a,t}, \quad 0 \leq p_{ij,dt}^{a,t} \leq 1 \quad (2)$$

where,

v_a^t	=	observed traffic count in link a during time interval t ,
T_{ij}^{dt}	=	trips leaving zone i to zone j during time interval dt ,
$p_{ij,dt}^{a,t}$	=	proportion of trips leaving zone i to zone j during time interval dt and traveling through link a during time interval t ,
i	=	origin,
j	=	destination,
a	=	link identifier,
dt	=	time interval for departure time, and
t	=	time interval.

Ashok (9) categorized the dynamic O-D estimation into closed network and open network. A closed network implies that complete traffic information of the network is available at all points in time. In real world, obtaining that kind of complete information can be rarely. On the contrary, an open network is common but it provides only partial information. The following section discusses several dynamic O-D estimation models for the open network found in literature. The models are categorized by solution techniques used in the dynamic O-D estimations in this section.

2.2.2.1 Generalized Least Squares (GLS) Estimator

Cascetta *et al.* (10) generalized and extended general statistical frameworks proposed for the static O-D estimations to the dynamic O-D estimation case. In their study, two approaches were proposed. The first approach, referred to as *simultaneous*, looks for an

estimator which gives, in one step, the whole O-D matrices for all time intervals by using traffic counts simultaneously as shown in Equation (3). The second approach, referred to as *sequential*, produces, at each step, estimates a O-D matrix for one time interval using two information: 1) traffic counts relating to the same time interval and to the previous time intervals and 2) O-D estimates relative to previous time intervals as shown in Equation (4). The *sequential* estimator has two merits over the simultaneous estimator: 1) the use of estimated O-D demands obtained from current time interval as a seed value for the next time interval and 2) the obvious computational advantage. The following equations are the general forms of the *simultaneous* estimator [Equation (3)] and the *sequential* estimator [Equation (4)].

$$(\mathbf{T}_1^* \dots \mathbf{T}_{n_t}^*) = \arg \min_{\mathbf{s}_1 \geq 0 \dots \mathbf{s}_{n_t} \geq 0} [f_1(\mathbf{s}_1 \dots \mathbf{s}_{n_t}; \mathbf{T}_1 \dots \mathbf{T}_{n_t}) + f_2(\mathbf{s}_1 \dots \mathbf{s}_{n_t}; \mathbf{v}_1 \dots \mathbf{v}_{n_t})] \quad (3)$$

$$(\mathbf{T}_t^*) = \arg \min_{\mathbf{s}_t \geq 0} [f_1(\mathbf{s}_t; \mathbf{T}_t) + f_2(\mathbf{v}_t, \mathbf{s}_t; \mathbf{T}_1^*, \mathbf{T}_{t-1}^*)] \quad (4)$$

where,

\mathbf{T}_h^* = vector of the estimated trips for all O-D pairs leaving the origin during time interval t ,

\mathbf{T}_t = vector of the initial true trips for all O-D pairs leaving the origin during time interval t ,

\mathbf{v}_t = vector of observed link flows at time interval t ,

\mathbf{s}_t = current value of the trips vector, and

t = time interval, $t = 1, \dots, n_t$.

Cascetta *et al.* (10) applied the above two Generalized Least Squares (GLS) estimators and tested the performances of their methods on the Brescia-Verona-Vicenza-Padua motorway in Italy (linear 140 km long network including 19 zones, 19 nodes and 54 links).

The accuracy of OD matrices obtained from these approaches heavily relied on the amount of available observed link flows (9). Furthermore, no normal procedure was developed for assigning dynamic demands onto the network to obtain the dynamic assignment parameters mentioned in the section [2.2.2]. However, the Cascetta et al.'s approach is a good initiative in that it provided explicit equations for modeling the dynamic O-D estimations using observed link flows (9).

Hellianga and Aerde (11) compared a least square error (LSE) model and a least relative error (LRE) model. In the LSE model, the best O-D estimate is the one that minimizes the sum of the squared absolute deviations between estimated and observed link flows [Equation (5)]. On the contrary, the LRE model determines the best O-D estimate as the one that minimizes the relative link error [Equation (6)].

$$\min \quad E = \sum_a (v_a - \bar{v}_a)^2 \quad (5)$$

$$\min \quad E = \sum_a \left[\ln \left(\frac{\bar{v}_a}{v_a} \right) \right]^2 \quad (6)$$

where,

- a = link identifier,
- \bar{v}_a = estimated flow on link a , and
- v_a = observed flow on link a .

Their study described the application of the above two models for a 35-km section of Highway 401 in Toronto, Canada. In their application, many field conditions including the true O-D matrix, the true routes, and the true route weights were unknown. Furthermore, loop detector data covered only 45% of the 35-km section of the freeway. The study results showed that the LRE model estimates showed a much higher correlation ($r = 0.95$) between observed and estimated link flows than that of the LSE model ($r = 0.75$). The study also argued that the LSE model tends to overestimate the link flows, particularly at low flows because it tends to place higher weight on links with large observed flows (11).

2.2.2.2 QUEENSOD

Van Aerde *et al.* (12) introduced the QUEENSOD model for generating dynamic synthetic O-D matrices and applied the QUEENSOD model to a 35-km section of Highway 401 in Toronto (12), Canada and Scottsdale/Rural Rd. and Hayden Rd. (an alternate parallel route to Scottsdale/Rural Rd.) network in Phoenix, Arizona consisting of 499 nodes and 1,021 links (36) and Salt Lake metropolitan region, Utah (3,365 nodes, 7,926 links and 565 zones) (37). The QUEENSOD model, which is based on an iterative approach, starts the first iteration from a seed O-D matrix, which can be a uniform or historic O-D matrix. The seed O-D matrix is utilized to generate estimates of link flow based on an estimate of drivers' expected route choices. The adjustment on the seed O-D matrix is conducted based on the quantitative comparisons between observed and estimated link flows. In this manner, the seed O-D matrix is systematically modified to produce a new O-D matrix. Detailed process will be explained in Chapter 4.

The QUEENSOD model is a computer program specifically oriented to satisfy practical traffic engineering as opposed to mathematical needs, while still providing a solution that is rooted in the state-of-the-art in a mathematical theory.

2.2.2.3 Kalman Filtering

Kalman filter algorithms (13) have been widely proposed to accommodate the real-time requirements (9, 14, 15). These algorithms solve least-square problems in an incremental fashion and allowing an update on the solution using measurement equation and transition equation when additional data is available (16). Okutani (14) presented a Kalman filtering-based model that estimates or predicts unobserved link traffic flows from observed link flows. The model includes an autoregressive formulation in which the state vector for a period is related by correlation factors to state vectors for prior periods [refer to Equation (7)]. Let us assume that $x_1(t)$ stands for an unobserved traffic flow on link 1 at time interval t , which is to be estimated from the observed flows on other links, $x_i(t)$ ($i = 2, 3, \dots, n$). According to Okutani (14), the following relationship (transition equation) holds between $x(t+1)$ and $x(t)$:

$$x(t+1) = \phi(t)x(t) + w(t) \quad (7)$$

where,

$x(t)$ = state vector of all link flows at time interval t ,

$\phi(t)$ = $n \times n$ transition matrix,

$w(t)$ = n dimensional noise vector with $E[w(t)] = 0$, $\text{cov}[w(t), w(s)] = R\delta_{ts}$,

R = matrix of size $n \times n$, and

δ_{ts} = Kronecker's delta.

The measurement (or observation) equation associated with the state vector $x(t)$ is given by:

$$y(t) = \theta x(t) + v(t) \quad (8)$$

where,

$y(t)$ = n-1 dimensional observation vector,

$v(t)$ = measurement error vector of size n-1 with

$$E[v(t)] = 0, \text{ cov}[v(t), v(s)] = R_1 \delta_{ts},$$

R_1 = (n-1)×(n-1) covariance matrix of $v(t)$,

θ = (n-1)×n transfer matrix of measurement system defined as
 $\theta = [O, I],$

O = zero vector of size n×1, and

I = n×n identity matrix.

The above Equation (8) states that the link flows except the flow on link 1 is observed. From Equations (7) and (8), we can obtain the estimate of $x(t)$ denoted by $\hat{x}(t)$ in accordance with the Kalman filter theory as:

$$\hat{x}(t) = \phi(t-I)\hat{x}(t-I) + K(t)[y(t) - \theta\phi(t-I)\hat{x}(t-I)] \quad (9)$$

where,

$$K(t) = \text{n} \times \text{(n-1) Kalman gain matrix.}$$

In Equation (9), the estimate $\hat{x}_1(t)$ is given by the first element of $\hat{x}(t)$.

Ashok (9) and Ashok and Ben-Akiva (15) formulated a Kalman filter based approach for real-time estimation and prediction. In order to overcome the inadequacy of Okutani (14)'s autoregressive specification for link flows, they introduced the notion of deviations of O-D demands from historical estimations. The state vector is hence defined in terms of O-D deviations that conform to an autoregressive process. The measurement equation is the same as Okutani's. The assignment fractions are obtained either by using the equations derived by Cascetta *et al.* (10) or by using a Dynamic Traffic Assignment (DTA) approach. The model was evaluated using actual traffic data from Massachusetts Turnpike, Massachusetts (120 miles, 15 entry/exit ramps, and 210 O-D pairs), a stretch of I-880 near Hayward, California (5.2 miles, 4 on-ramps, 5 off ramps and 20 O-D pairs) and a freeway encircling the city of Amsterdam, Netherlands (32 km, 20 entrance and exit ramps (9).

Hu (35) also introduced Kalman filtering algorithm in dynamic O-D estimation and prediction problem in which a similar autoregressive process to Ashok and Ben-Akiva (15) was used in the transition equation. The approach was evaluated at a hypothetical freeway network, which consisted of 8 zones, 32 nodes and 70 links, and historic O-D matrix. All entrance, exit and mainline traffic counts for the approach were obtained from DYNASMART simulation results. In the approach, Hu included a control equation within the Kalman filtering algorithm to consider drivers' dynamic responses to traffic conditions (i.e., their route change according to traffic information or congestions). For the equation, a discrete choice model was included to calculate time-varying route switching probabilities of each driver for given travel time and different routes (35).

The advantage of Kalman filter algorithm is its ability of accommodation for estimation and on-line prediction of dynamic O-D matrix (15, 35). The main drawback of the Kalman filter algorithm appears to be its inability to handle large-scale O-D estimation problems. The analytical computation of the normal equations and the variance propagation require intensive linear algebra computations. Moreover, the sparsity of the least-square problem (the amount of available observed traffic counts) is not exploited by the algorithm and a lot of fill-in has to be performed (16). However, when traffic conditions are normal, or when the time intervals are short, the auto-regressive process can provide a good estimate of the O-D matrix (16).

2.2.2.4 Maximum Likelihood Estimation (MLE)

He *et al.* (17) proposed a Maximum Likelihood Estimation (MLE) approach to estimate the parameters of dynamic O-D demand and route choice simultaneously. In order to derive a full likelihood functions for dynamic O-D and route choice, they presented approximate joint probability distribution function of the temporal link flows on a network. This algorithm was tested in two simple and ideal networks: i) a network including 4 nodes and 5 links and ii) a network consisting of 9 nodes and 12 links. The proposed model could incorporate prior information. It can estimate the parameters with measurement errors and incomplete data.

2.2.3 Estimation of Temporal Route Choice for Dynamic O-D Estimation

The most important input in the dynamic O-D matrix estimation is the calculation of temporal route choices of individual drivers. In static O-D matrix estimation, this route

choice (assignment) is broadly classified into two main groups: proportional and non-proportional assignments. Proportional assignment method makes the proportion of drivers choosing each route independent from flow levels. The all-or-nothing assignment is an example of this kind of assignments. Non-proportional assignment explicitly considers congestion effects such that the proportion of travelers using each link does depend on link flows. Equilibrium or stochastic assignments are used in the non-proportional assignment method.

However, to estimate dynamic O-D matrix, the assignment has to consider temporal variation of congestions. Hence the traditional assignment method is not sufficient for this kind of estimation. In order to obtain temporal route choices, DTA or traffic simulation models are usually used (9, 12, 15, 18). For example, QUEENSOD model is linked with INTEGRATION, a microscopic traffic simulation model.

2.3 GENETIC ALGORITHM

In this section, genetic algorithms are investigated. Genetic algorithms (GA) were developed by John Holland in the early 1970s at the University of Michigan (19). GA is a family of computational models inspired by evolution and natural selection (20). These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. It can be classified as a guided random search algorithm to optimize any form of objective function (21).

2.3.1 Introduction

Genetic algorithms have been used to solve problems with objective functions that are difficult to work out with mathematical approaches (19, 22, 23). In particular, lots of transportation systems are based on stochastic systems, which are difficult to be represented in mathematical formulations. Dynamic O-D matrix estimation is a good example. In Equation (1), the observed traffic count (v_a) in a particular link a is the summation of the contributions of all trips between zones using that link. The trips are determined according to the dynamics of vehicle-to-vehicle, vehicle-to-control system, congestions and resulting route choices. Thus, it is not easy to represent the resulting trips with a mathematical formulation.

Genetic algorithms maintain and manipulate a population of potential solutions and implement a “survival of fittest” concept to search better solutions. The GA also provides an implicit as well as explicit parallelism (24). Explicit parallelism allows the exploitation of several promising areas of the solution space at the same time through generations. The implicit parallelism is due to the schema theory developed by Holland. Schema theory is explained in the Sections 2.3.2.5.

2.3.2 Mechanisms of Genetic Algorithm

The genetic algorithm is a population-based model that uses selection and recombination operators to generate new sample points in a search space. In general, the fittest individuals in the population tend to reproduce and survive to the next generation, thus improving the quality of successive generations. Genetic algorithms have been shown to

solve linear and nonlinear problems by exploring all regions of search space and exponentially exploiting promising areas through selection, crossover and mutation operations (25).

2.3.2.1 Solution Representation

In GA, each individual solution should be systemically described by a chromosome-like data structure. The chromosome-like data structure is made up of a sequence of genes from a certain alphabet. An alphabet could consist of binary digits (0 and 1), floating point numbers, integers or symbols (24).

1	0	1	0	0	0	1	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Figure 1. Example of Chromosome-like Data Structure Using Binary Digits

The Figure 1 illustrates an example of chromosome-like data structure using binary digits. This solution representation consists of three parameters (or genes). The first parameter (the first four binary digits) means '12' in integer value.

The representation scheme determines how the problem is structured in the GA and also selects the genetic operators that are used. Problem representation (or solution representation) has been the subject of much investigation. It has been shown that more natural representations are more efficient and produce better solutions (25).

2.3.2.2 Selection Function

The selection function of individuals to produce successive generations plays an extremely important role in the genetic algorithm. A probabilistic selection is performed based on the individual's fitness such that the better individuals have an increased chance of being selected. An individual in the population can be selected more than once. There are several schemes for the selection process: roulette wheel selection and its extensions, scaling techniques, tournament, and ranking methods (23, 25).

Roulette wheel method, developed by Holland (19), works as follow: The probability of selection for each individual i , P_i , is defined by:

$$P_i = \frac{F_i}{\sum_{j=1}^{PopSize} F_j} \quad (10)$$

where,

P_i = probability where individual i is chosen, and

F_i = fitness of individual i .

The use of roulette wheel selection limits the genetic algorithm to maximize the evaluation function. Extensions, such as windowing and scaling, have been proposed to allow for minimization (24). Ranking method is also used and it requires the evaluation function to map the solutions to a partially ordered set to allow for minimization.

Ranking method assigns P_i based on the rank of solution i when all solutions are sorted.

Normalized geometric ranking (24) defines P_i by:

$$P_i = q'(1-q)^{r-1} \quad (11)$$

where,

$$\begin{aligned} q &= \text{the probability of selecting the best individual,} \\ r &= \text{the rank of the individual } i, \\ N &= \text{population size, and} \\ q' &= \frac{q}{1-(1-q)^N} \end{aligned}$$

Tournament selection also requires the evaluation function to map solutions to a partially ordered set. However, it does not assign probabilities. Tournament selection works by selecting j individuals randomly, with replacement, from the population, and inserts the best of the j into the new population. This procedure is repeated until N individuals have been selected.

The elitist selection method keeps the best individual from generation to generation. If the best individual has not been transferred to the next generation during the reproduction, crossover and mutation processes, the elitist method copies the best individual from the current population to the next population to ensure its survival (24). This elitist selection method is usually used with other selection methods.

2.3.2.3 Genetic Operators

Genetic operators provide the basic search mechanism of the genetic algorithm. The operators are used to create new solutions based on existing solutions in the population. There are two basic types of operators: crossover and mutation (24). The crossover takes

two individuals and produces two new individuals while the mutation alters one individual to produce a single new solution.

The application of these two basic types of operators and their derivatives depends on the solution representations (binary, real-valued number or symbol). In this section, simple mutation and simple crossover processes for binary solution representation are explained.

Let x_i and y_i be two m -dimensional row vectors denoting individuals (parents). Simple binary mutation generates a random number r from a uniform distribution and flips each bit in every individual in the population with probability p_m according to the Equation (12).

$$x'_i = \begin{cases} 1 - x_i, & \text{if } r \approx U(0,1) < p_m \\ x_i, & \text{otherwise} \end{cases} \quad (12)$$

Simple crossover generates a random number r from a uniform distribution and creates two new individual (x'_i and y'_i) according to the Equation (13).

$$x'_i = \begin{cases} x_i, & \text{if } r \approx U(0,1) < p_c \\ y_i, & \text{otherwise} \end{cases} \quad (13)$$

$$y'_i = \begin{cases} y_i, & \text{if } r \approx U(0,1) < p_c \\ x_i, & \text{otherwise} \end{cases}$$

2.3.2.4 Initialization and Termination

The genetic algorithm must start with an initial population. The most common method is to use randomly generated solutions for the entire population. However, since GAs can iteratively improve existing solutions, the beginning population can be seeded with one potentially good solution with the remainder of the population being randomly generated.

The GA moves from generation to generation by selecting and reproducing individuals until a termination criterion is met. The most common stopping criterion is a maximum number of generations. Another termination strategy involves population convergence criteria. In general, GAs will force much of the entire population to converge to a single solution. When the sum of the deviations among individuals becomes smaller than some specified threshold, the algorithm can be terminated. The algorithm can also be terminated due to a lack of improvement in the best solution over a specific number of generations. Several strategies can be used in conjunction with each other.

2.3.2.5 Schema Theorem

Even though there exists no established general theory that explains exactly why genetic algorithm works, several hypotheses have been proposed which can partially explain the mechanism of genetic algorithms (27).

The basic theory of genetic algorithms is based on a binary string representing a solution, and on the notion of schemata (25). A schema is a string of total length l , taken from the symbols $\{0, 1, *\}$, where, ‘*’ is a ‘do not care’ symbol. A schema represents the set of all binary strings of length l , which match with all positions other than ‘*’. For

example, consider the schema, * 1 0 *. It matches with four strings, {0100, 0101, 1100, 1101}. It is noted that every schema matches exactly 2^r strings, where r is the number of ' * ' in a schema.

Two important properties of schema are their order and defining length. The schema theorem is built from these properties. Order is the number of fixed bit-positions (i.e., 0 and 1 position) in a schema. Defining length is the distance between the first and last fixed bit positions.

The schema theorem (19) was the first rigorous explanation of how genetic algorithms work. It says that “short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetics algorithm.” Michealewicz (25) explained the schema theorem in more detail with the calculation of survival probabilities

2.3.3 Genetic Algorithms in O-D Estimation

A few studies in the area of the static O-D matrix estimation have been successfully conducted with GAs. Kim *et al.* (6) proposed the GA to estimate static O-D matrices. In the study, the sensitivity analysis based (SAB) algorithm and the GA as solution algorithms based on a bilevel programming approach were compared using a simple network consisting of nine nodes and fourteen links. The objective function is formulated as minimization of square errors based on the bilevel program approach proposed by Yang (28). The objective function consists of traffic counts and historic O-D matrix as shown in the Equation (14).

$$\min Z(\bar{T}_{ij}) = \frac{1}{2} \sum_{a \in A} (\bar{v}_a(\bar{T}_{ij}) - v_a)^2 + \gamma \frac{1}{2} \sum_{ij \in W} (T_{ij} - \bar{T}_{ij})^2 \quad (14)$$

where,

\bar{T}_{ij} = estimated trips between zone i and zone j , $ij \in W$,

T_{ij} = historical trips between zone i and zone j , $ij \in W$,

\bar{v}_a = estimated link flows on link a , $a \in A$,

v_a = observed link flows on link a , $a \in A$,

a = link identifier,

W = vector for all zone pairs, and

A = vector for all link.

Here, γ is a parameter reflecting the reliability of the historical O-D matrix. If the O-D matrix structure is altered by changes in land uses, a new traffic facility or etc., γ should have lower value since there is no significant dependence between the historical O-D and the current O-D patterns. In order to obtain the assignment parameter (or driver's route choices and its weight), equilibrium assignment was used. The individual solutions were represented with two chromosomes:

$X_n[m][ij]$ = choice proportion of m^{th} chromosome between i and destination j in n generation, where $m=1, 2, \dots, M$, M is the number of population, and

$Y_n[m][ij]$ = proportion of m^{th} chromosome from origin i in n generation.

Using the two chromosomes, the GA calculates the estimates of trips as follow:

$$t_n[m][ij] = X_n[ij] \cdot Y_n[m][ij] \cdot O_i \quad (15)$$

where,

O_i = historical trips generated from origin i .

Using the calculated trips and resulting link flows from equilibrium assignment, the fitness values of individual solutions are calculated from the Equation (14). Real-valued GA was used with arithmetic crossover and uniform mutation (24). The maximum generation number was used as a stopping condition.

Yin (7) also used GA-based approach to solve a bilevel programming model for static O-D estimation with two simple networks. The basic idea of the GA approach is to code the decision variable (trips between zone-pairs) of the upper-level problem (minimizing the difference between historical O-D matrix and estimated O-D matrix) and calculate the fitness of each chromosome by solving lower-level problem (minimizing the difference between observed link flows and estimated link flows). In order to solve the bilevel problems, Binary GA was used with single-crossover and single-bit point mutation (simple mutation) and tournament selection. The maximum generation number was used as a stopping condition.

From the numerical tests, the GA-based approach based on the global perspective and implicit parallelism was founded to be more efficient than to SAB algorithms for the static O-D estimation. Therefore, the GA-based approach could estimate the current O-D matrix correctly by using the historic O-D matrix, which was surveyed before changes in trips patterns because the GA decreases the dependency on initial values. However, the

GA-based approach required more computation efforts than the SAB algorithms but avoids the complex computation.

2.4 TRAFFIC SIMULATION MODEL IN ITS EVALUATION

In this section, researches that used microscopic traffic simulation models in evaluation on ITS applications are summarized.

Nelson and Bullock (39) examined the impact of emergency vehicle preemption on closely spaced arterial traffic signals on State Route 26, Indianan. In their study, CORSIM was used to gain the quantitative data according to control and preemption logic from three traffic signal controllers linked to CORSIM for the SR-26 arterial.

Hansen *et al.* (40) connected the adaptive signal control system SCOOT to CORSIM to evaluate the performance of the adaptive signal control system in a 6 nodes network. In their approach, CORSIM provides the necessary traffic detector data for SCOOT optimization such that SCOOT can be evaluated under various traffic conditions generated by the traffic simulation model.

Lucas *et al.* (41) evaluated the performance of the RHODES (real-time hierarchical optimized distributed effective system) under various traffic environment generated by CORSIM. Here, CORSIM produced second-by-second traffic detector information and evaluated the function of the traffic control of the RHODES via communication using a dynamically linked library (DLL).

Chu *et al.* (43) presented a micro-simulation method to evaluate potential ATMIS applications using PARAMICS. In the study, PARAMICS provided control of traffic

operations and realized several functionalities of ATMIS strategies using APIs under incident scenarios in a corridor network located in Irvine, California.

2.5 SUMMARY

The above review has summarized the various approaches for dynamic O-D matrix estimations from traffic information and it certainly illustrates the complexity of the problem. Several techniques including generalized least squares (GLS) estimation, QUEENSOD model, maximum likelihood estimation (MLE) and Kalman filtering have been used for finding solutions for the dynamic O-D estimation problem in previous studies. Each technique has its own strength and weakness related to i) the amount of available traffic information, ii) capability for a large-scale network and iii) finding reasonable drivers' route choice and its weight.

CHAPTER 3

A DEVELOPMENT OF LARGE SCALE CASE STUDY SIMULATION NETWORK

3.1 INTRODUCTION

With the advances in the computation technology and needs for better evaluation tools in the design and analysis of transportation network, the use of microscopic simulation programs has been widely practiced. This is, in part, because the microscopic simulation tools provide inexpensive, fast, and risk-free evaluation environment.

However, when researchers and practitioners are to use a microscopic simulation program in their studies, they often face that certain features are not provided with the original program or not sufficient in achieving desired evaluations. For example, the implementation of coordinated actuated signal timing control logic in the PARAMICS program is very limited. One solution to these is adding a user-specified module into the program using an application programming interface (API). This has been well exercised by researchers but very limited to practitioners.

The purpose of this chapter is to develop a large-scale case study simulation network. This chapter expresses the detailed process related to building a basic network and developing an API for coordinated actuated control logic in PARAMICS. As a case study, this study develops a simulation-based large-scale test-bed for the City of Hampton, Virginia, in the U.S.A. using a microscopic simulation package, PARAMICS. Figure 2 shows the location of City of Hampton in Hampton Roads, VA.



Figure 2. City of Hampton, Virginia

3.2 MICROSCOPIC SIMULATION MODEL SELECTION

According to Rakha *et al.* (37), a large network is defined as a network with more than 1,000 links. They described the requirements of a validated microscopic model for large-scale modeling as followings:

“ i) the model must be capable of modeling origin-destination demand tables, ii) the model must be capable of modeling dynamic traffic routing, and iii) the model must be capable of modeling the dynamic interaction of freeway/arterial facilities.”

In order to search appropriate microscopic traffic simulation model, we compared CORSIM, SYMTRAFFIC, VISSIM and PARAMICS in terms of capability of large-scale network and the above three requirements (3). As a result of that, we selected PARAMICS for this study. In PARAMICS, there are no logical limitations on the number of links, nodes and vehicles whereas CORSIM and SYMTREAFFIC cannot support any large-scale network due to limitation on network size. VISSIM and PARAMICS can support O-D matrix-based assignments. In particular, PARAMICS provides various assignment methods including all-or-nothing, stochastic assignment, dynamic feedback assignment and their combinations whereas VISSIM allows one dynamic assignment method. These various assignments of PARAMICS can support to investigate various route choice behaviors of individual drivers. In addition, PARAMICS showed faster computational ability than VISSIM as shown in Table 1.

Table 1. Summary of Simulation Run Times Comparison

Programs	Simulation Conditions	Average (Second)	Median (Second)	Standard Deviation
VISSIM	With animation, with VAP	300.8	298.0	6.76
	Without animation, with VAP	203.2	202.0	4.6
	With animation, without VAP	242.7	244.0	4.16
	Without animation, without VAP	145.2	145.0	0.84
PARAMICS	With animation, with API	296.3	295.5	4.13
	Without animation, with API	42.3	40.0	6.31
	With animation, without API	298.0	298.0	1.87
	Without animation, without API	41.6	40.0	6.19

- 1) These comparisons of simulation run time were conducted in the network including four intersections that is operated in coordinated actuated signal control, and
- 2) These comparisons of simulation run time were conducted in the computer with Intel Pentium III (933MHz) and 128MB memory.

3.3 MICROSCOPIC SIMULATION MODEL, PARAMICS

PARAMICS, developed by QuadStone Ltd., in U.K., is a suite of high performance software tools that provides microscopic, time-stepping, and scalable traffic simulation (29). The complete suite of the PARAMICS software comprises five modules: i) Modeler, ii) Processor, iii) Programmer, iv) Analyzer and v) Monitor. Among these, the Modeller is a core simulation and visualization tool and the Programmer is an API to PARAMICS (30). PARAMICS has been chosen by traffic researchers and practitioners in the US for their traffic studies.

3.3.1 User Interface

The PARAMICS model provides a graphical user interface to build networks and to watch the simulation results (Analyzer) and animation (Modeller). User can build the layout of network using background images in DXF, BMP, and OS/NTF Strategi format.

3.3.2 Network Structure

The network in PARAMICS is based on node-link structure. Node usually represents the intersection or the point where attributes of link connecting the node to other nodes change. Link means a road street. PARAMICS defines the route of each vehicle through assignment. Thus, PARAMICS has zones and zone connector. Each vehicle starts its trip from origin zone to destination zone. Zone connector links a zone and a link in a network. The zone connector has zero impedance and is not considered in calculation of MOEs such as travel time.

3.3.3 Traffic Generation and Assignment

Travel demand in PARAMICS is defined as an origin-destination matrix of origin. The trips are proportioned into vehicle type and are profiled by 5 minute time period for a maximum of 24 hours. Based on this time period and demand, vehicles are randomly released.

The choice of time period is governed not only by the fluctuation in time dependent traffic demand but also network changes such as different traffic signal settings by time of day. This gives the user a lot of flexibility in the design. The three

main assignment techniques used in PARAMICS are all-or-nothing, stochastic and dynamic feedback assignments. Stochastic assignment method accounts for variability in travel cost (or driver's perception of those costs). This method assumes that the link costs perceived by drivers can be different depending on their characteristics. Dynamic feedback assignment assumes that drivers who are familiar with the road network will re-route if information on the present state of traffic condition is fed back to them.

3.3.4 Signal Control in PARAMICS

The built-in traffic signal control logic in PARAMICS is designed for pretimed signals under single ring structure. Thus, the advanced signal control logics such as actuated signal control need to be developed. This can be achieved in two ways: an API and a plan language. The API has to be coded in a C++ language, while a built-in plan language can be written as a script.

3.3.5 API in PARAMICS

The PARAMICS Programmer is a framework that allows advanced users to customize existing control logics in the simulation model using APIs. The user customized logics are implemented through a dynamic link library (DLL), which is generated from a C++ file containing the customized logic. A few examples of such API applications have been already exercised. These include signal control logic (3, 31) and parameter calibration (32).

3.3.6 Model Weaknesses

Although PARAMICS allows the API and the plan language to develop coordinated actuated signal control logic, it is not easy for practitioner end users to build one. In addition, PARAMICS does not allow the dual-ring concept or NEMA control concept. Furthermore, PARAMICS doesn't have an automatic diffusion function. For example, in PARAMICS if a vehicle were unable to make lane change it would stop and block other vehicles. This leads to discrepancies in performance when compared to real world condition and causes higher variability in simulation output. With a diffusion feature, a vehicle would not wait more than user-defined diffusion time. The vehicle would be removed from the simulation after diffusion time.

3.4 PARAMICS NETWORK

The PARAMICS network, as shown in Figure 5, consists of 50 zones, 3,364 links, 1,464 nodes, and 154 signals. The 97 intersections out of the total 154 signals are being operated under a coordinated and actuated mode.

3.4.1 Network Building

Network building for this study was conducted as the following process (29):

- 1) Preparing Data.
 - Background image of the City of Hampton
 - Distance between two specific points within the desired network

- Categories of link types in terms of number of lanes, speed limits, arterial or freeway, major or minor, and width of median
 - Detailed geometries of intersections and ramps
 - Control types of intersections and their signal timing plans
 - O-D matrices
- 2) Importing background image and adjusting scale parameters for the imported image using the distance measured in advance.
 - 3) Adding nodes for intersections and ramps using the Network Editor.
 - 4) Adding links between nodes using the Network Editor.
 - 5) Making network layout to be realistic through adding dummy nodes and links or using Curve Editing function in the Network Editor.
 - 6) Installing zones using the Network Editor.
 - 7) Entering link attributes using Link Attribute Window in the Network Editor.
 - 8) Fine-tune geometries of intersections using the Network Editor
 - Installing left turning bay, right turning bay, and detectors
 - Changing location of curbs and stop bars
 - 9) Entering signal timing plans to each intersection using Edit Junction Window in the Network Editor.
 - 10) Entering O-D matrix.
 - 11) Setting simulation parameters.
 - 12) Watching animations.
 - 13) Repeat the above steps if necessary.

The layout of the PARAMICS network is coded based on a bitmap image from a Geographic Information Systems (GIS) map (Figure 3). Using the GIS map of the City of Hampton, the distances between intersections were determined.

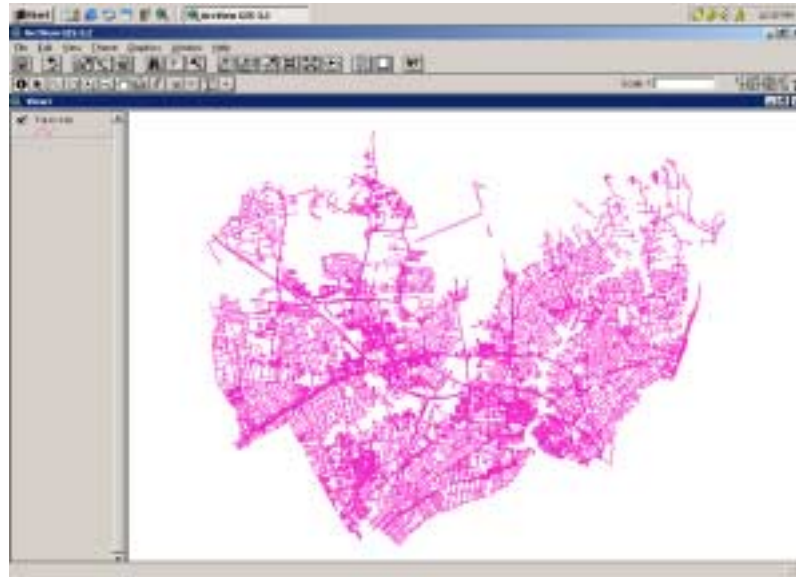


Figure 3. GIS Map Used as Reference for Network Coding

The link attributes (number of lanes, geometry, link type – major, minor and freeway, etc.) and the node attributes (intersection geometry, intersection shape, number of lanes, lengths of left turning and right turning bays, etc.) were gathered from images in Lizard Tech's MrSID format as shown in Figure 4.



(a) Zoomed-out Image



(b) Zoomed-in Image

Figure 4. Images in Lizard Tech's MrSID Format

Through the process mentioned in the above, the following traffic simulation network including 50 zones, 3,364 links, 1,464 nodes and 154 signals was finally built.

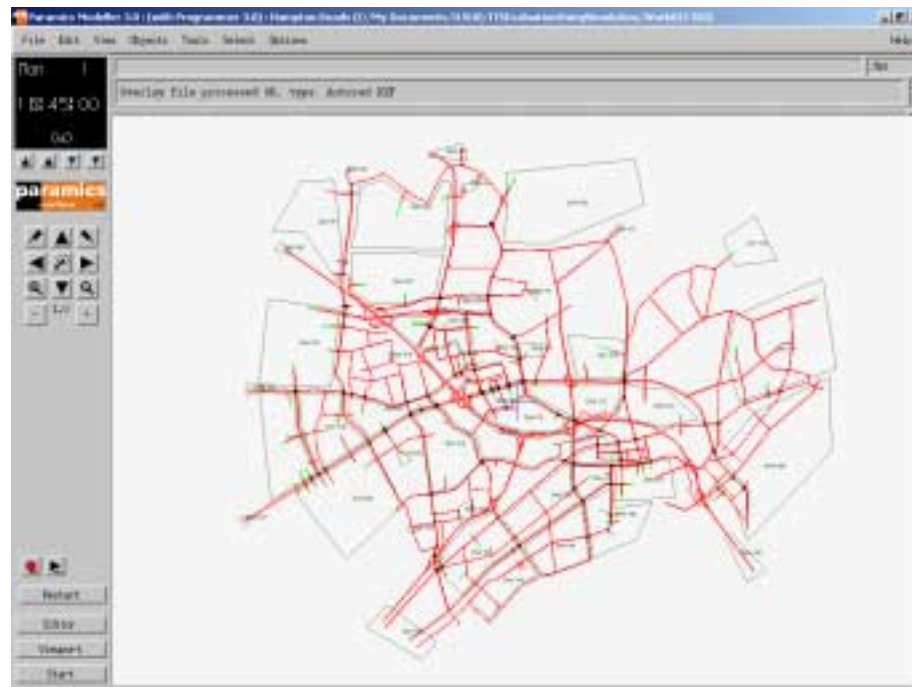


Figure 5. PARAMICS Network for the Case Study

3.4.2 Signal Control Logics

An API for the actuated and coordinated signal control logic was developed. The API code consists of Visual C++ code of about 3,700 lines. The signal control logic can realize actuated control features including gap-out, force off and phase skip. An API code can control all intersections in the network. The development of this process is detailed in section [3.4].

3.4.3 Parameter Setting

The network started with default parameter values. Default values of 1 second each were used for the most important network wide calibration parameters: mean headway and mean reaction time. It is noted that in order to avoid unrealistic simulation the lengths of the entire links were used as signposting distances (i.e., the distance from the intersection that drivers become aware of the intersection). In PARAMICS, if a proper signposting distance is not provided, a vehicle that needs to make lane change occasionally blocks the traveling lane until it finds acceptable gap to make lane change. Even with the maximum signposting distance, extremely large queues and network gridlocks were observed. In order to reduce such unrealistic behaviors, the dynamic feedback time is reduced to 120 seconds from default of 350 seconds.

3.4.4 Simulation

Since microscopic simulation programs start from empty network, they need initialization period. This basically fills the network such that the network condition and the simulation

output become more realistic. It is decided to use 60 minutes of network warm-up time and 60 minutes of simulation time. Default time step of 0.5 seconds is used such that each vehicle's position is updated every 0.5 seconds. There exist multiple paths among O-D zones in the test network. Thus, a dynamic assignment method in PARAMICS was used so that the variability in travel costs (or driver's perception of those costs) plays the primary role in finding optimal routes.

3.4.5 Visualization

The visualization involves the observation of the simulation runs and its purpose is to ensure that the traffic is moving through the network in a realistic manner. Some corrections were made to adjust specific behaviors at links, on lanes or at intersections via this visualization. These corrections include moving curb locations, stop line control points, forced lane change positions, etc (29). In particular, “next lane” for nodes and “signpost distance” for links are used for forced lane change by replacing default PARAMICS lane changing values. Setting “next lane” in a node and adjusting the parameters for “signposting” were critical in achieving realistic simulation. Parameters such as “moving curb” and “stop line control points” were used for smooth and realistic movements of individual vehicles at specific locations. These four parameters were changed after observing the visuals from the simulation.

Figures 6 and 7 show the impacts of inappropriate “next lane” setting and parameters for “signposting.”

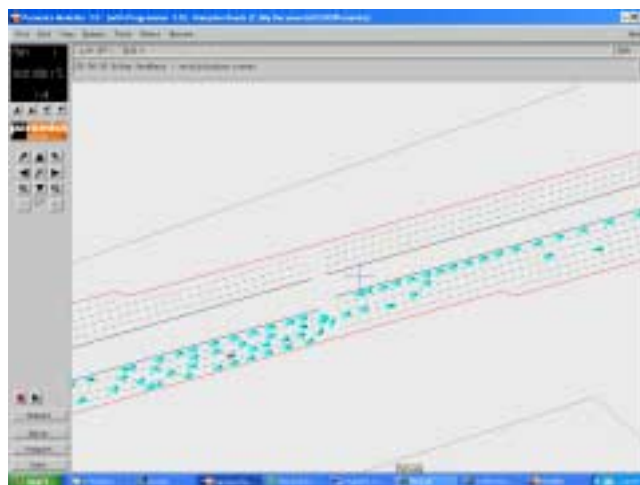


Figure 6. Example of inappropriate “next lane”

As seen in Figure 6, all vehicles moving from the upstream link to the downstream link use only the third lane. This abnormal behavior could occur in a regular network if the “next lane” parameter is not set properly.

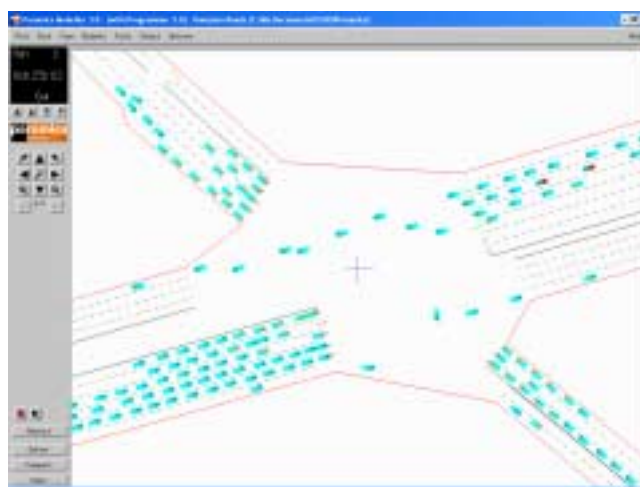


Figure 7. Impact of Insufficient Distance for “signposting”

Figure 7 shows some vehicles blocking the through lanes. These vehicles need to make left turns but are unable to make appropriate lane changes due to lack of acceptable gaps. This situation may lead to distortion in simulation results as the vehicles that are not on their proper lanes can lead to huge unrealistic congestion. This can be avoided by increasing the distance of “signposting”.

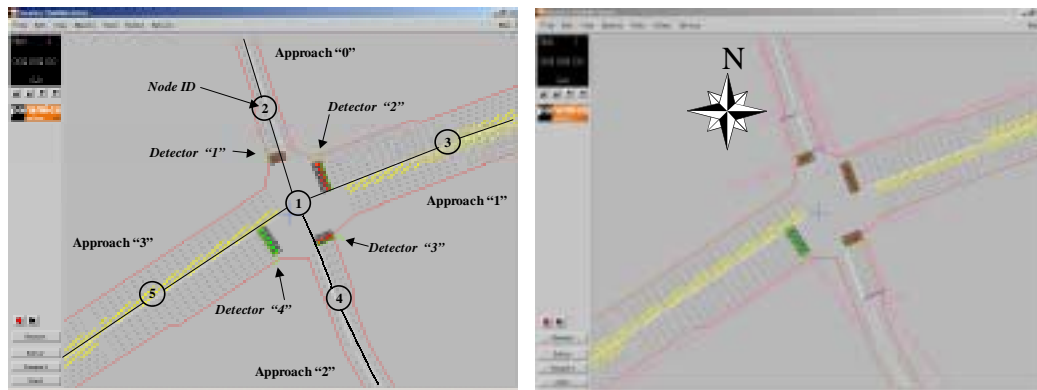
3.5 DEVELOPMENT OF AN API FOR COORDINATED ACTUATED SIGNALS

A major disadvantage in a pretimed signal is that its phase times do not respond to fluctuating traffic volume conditions. Unlike the pretimed signal, an actuated signal can adjust the phase times according to the prevailing traffic conditions (33). Furthermore, coordinated actuated signals can provide progression by maintaining coordinated phase times and allow phase skip and gap-out on non-coordinated phases. Thus, unused phase green times are added to the coordinated phases (34).

The API developed in this paper realizes coordinated actuated signals with 25 possible types of phase sequences (refer to the Appendix A). In the process of the API development, this research used PARAMICS Modeler v.3, Programmer v.3., and Microsoft Visual C++ v.6.

3.5.1 Geometry of Intersection and Detectors

Figure 8 illustrates the layout of a four-leg intersection being operated under coordinated actuated control logic. The eastbound and westbound approaches are coordinated phases.



(a) Layout of Intersection

(b) Signpost

Figure 8. An Example Intersection in the PARAMICS Network

Detectors are essential components in the operation of coordinated and actuated signals. A detector in PARAMICS can cover entire lanes on an approach link where detector is installed (29). The proposed API is designed to utilize up to three detectors on each approach. The specific usage of detectors is explained as follow.

One detector case: One stop bar detector determines both the presence of through and left turn vehicles during red interval and the extension of the through movement during green interval.

Two detectors case: One stop bar detector determine only the presence of through and left turn vehicles during red interval, while the other upstream detector (second one) determines the green extension of through movement during green interval.

Three detectors case: Two stop bar detectors determine the presence of each of through and left turn vehicles during red interval, while the upstream detector (third one) determines the extension of the through movement during green interval.

The functionality of detectors may be changed according to the hierarchy of approach (coordinated approach or uncoordinated approach) or traffic controller type (fully actuated or semi actuated). A recent study (31) suggested that the use of multiple detectors instead of a long loop on left turning lanes. The study also noted that simulation time would increase with the use of multiple detectors as PARAMICS calculates the status of entire detectors in every time step.

3.5.2 Yellow Change Interval

In PARAMICS, yellow change interval time is a global variable such that only one value is used for the entire phases in the network. Furthermore, yellow change interval is not recalled for the phase once a phase skip occurs. Thus, the yellow change interval and the actual green interval are combined such that only effective green interval (1) is used. PARAMICS does not provide any functions that can control yellow change interval. Only green time and all red time are controllable by “signal_action” call back function (31). Because of this limitation, yellow change intervals are added to the green intervals in this study as shown in Table 2.

3.5.3 New Phase Sequence for API

PARAMICS uses a phase representation that is different from a dual ring, eight-phase controller. Figure 9 shows the splits and phasing diagram of Synchro (i.e., traffic signal coordination software) for the phase sequence number 1 (refer to Appendix A). The splits and phasing diagram of SYNCHRO are a graphical representation of the current splits and phasing based on a dual ring, eight-phase controller (34).



Figure 9. Splits and Phasing Diagram Based on a Dual Ring, Eight-Phase Controller

A phase symbol ' ϕ ' and the NEMA phase number is shown next to the movement symbol (arrows in Figure 9), while green split time in seconds is shown in the bar chart right below the movement diagram. The green split time includes green interval and intergreen times (yellow plus all red interval). However, PARAMICS uses a phase representation based on a single ring controller for its built-in pretimed controller. The single ring controller has the phases to operate one at a time sequentially as shown in Figure 10.

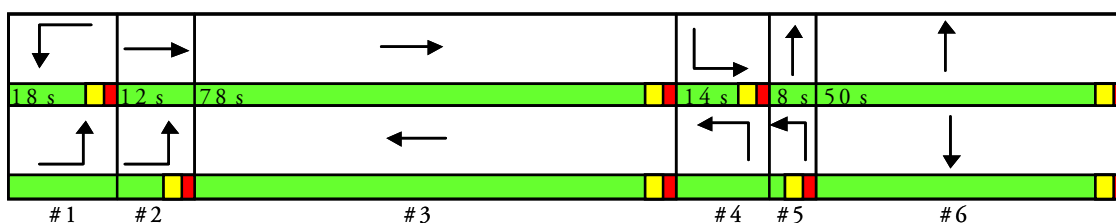













Figure 10. Splits and Phasing Diagram Based on a Single Ring Controller in PARAMICS

A phase number in PARAMICS is identified with the phase symbol # as shown in Figure 10. The phase number identifies a group of non-conflicting movements arranged in an established order of preferred sequence. For the dual ring controller (see Figure 9), the phase $\phi 1$ identifies the westbound left turn movement. For the PARAMICS single ring controller (see Figure 10), phase #1 means the movement of the westbound and eastbound left turns. The phase #2 is an overlap phase here. If an overlap phase exists, PARAMICS gives no intergreen (yellow and all red time) time between consecutive movements. For example, when PARAMICS changes green time from phase #1 to phase #2, it assigns intergreen time only between the westbound left turn and eastbound through as shown in Figure 10. This is because the eastbound left turn on phase #1 and #2 is consecutive phase. Each phase in Figure 10 can be individually skipped and gapped-out if no demand is present except for the coordinated phases. In order to implement phase skip and gap-outs in PARAMICS, this study uses a few dummy phases as shown in Table 2.

Table 2. Proposed Phase Sequence for API in PARAMICS

Times (second)	Phases												
													
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13
Green	75	11	-	-	5	-	47	15	-	-	9	-	-
Yellow****	-	-	-	-	-	-	-	-	-	-	-	-	-
All Red	3	3	-	-	3	-	3	3	-	-	3	-	-
Marks*	A	A	D	D	O**	D	A	A	D	D	O**	D	D

*“A” means an actual phase, “D” means a dummy phase, and “O” means an overlap phase.

** These overlap phase is specified as a dummy phase if there exists no overlap phase.

*** The Phase #1 for these through movements is assumed as a coordinated phase.

**** Yellow times are included into Green interval as mentioned in the above

In Table 2, these dummy phases are used to realize the revised phase sequence due to gap-out, phase skip or extension on coordinated phase. For example, if no demand were present for phase #2 (both northbound and southbound left turns), the proposed API would skip the phase #2 and serve phase #7. If no demand for only northbound left turns were present for phase #2, the API would skip the phase #2 and serve phase #3 followed by phases #4 and #7. Here, the phase #4 works as an intergreen time for the southbound left turn movement and it serves between new phase #3 and actual phase #7. The phase sequence starts from the coordinated through movements (eastbound and westbound through movements) in order to keep the offset reference points. This is because it appears that the offset reference point in PARAMICS is always the start of the first phase (phase #1).

3.5.4 Input Data Structure

Two input data files are required to implement the proposed API. One is the “priorities” file – a standard input file in PARAMICS. The “priorities” file includes information like green interval, maximal allowed green interval, all red time and movement categorizations (i.e., major, medium and minor) for each phase. The other required input file contains the “intersection layout and signal timing” data. It includes detailed information about intersection layout, NEMA phase numbering, signal timing plan and etc. Detail descriptions are shown below.

3.5.4.1 “priorities” file

The “priorities” file defines green and red time interval and the hierarchy of movements.

The following is an example of the first phase in the API (through movements on major approaches):

<i>actions 1</i>	<i>(node ID for this intersection)</i>
<i>phase offset 0.00 sec</i>	<i>(offset)</i>
<i>phase 1</i>	<i>(phase number)</i>
70.00	<i>(green interval)</i>
max 135.00	<i>(maximal allowed green interval)</i>
<i>red phase 3.00</i>	<i>(all red time)</i>
<i>fill</i>	<i>(no use of yellow time)</i>
<i>all barred except</i>	<i>(priority of movements)</i>
<i>from 2 to 5 minor</i>	<i>(setting turning movement of 2 → 1 → 5 as minor)</i>
<i>from 3 to 2 minor</i>	<i>(setting turning movement of 3 → 1 → 2 as minor)</i>
<i>from 3 to 5 major</i>	<i>(setting turning movement of 3 → 1 → 5 as major)</i>
<i>from 4 to 3 minor</i>	
<i>from 5 to 3 major</i>	
<i>from 5 to 4 minor</i>	

The hierarchy of priorities follows the order of “major”, “medium”, “minor”, and “barred.” “Major” priority movements are free flow and not restricted by other streams of traffic. “Medium” priority gives way (yields) to “major” streams of traffic but has priority over “minor” traffic movements. “Minor” priority yields its right of way to both “major” and “medium” traffic flows while “barred” indicates the turn is banned to all vehicle movements (1). In this study, “major”, “medium” and “minor” priorities are used for protected through or left movement, permissive left movement and right turn respectively. For dummy phase, no green interval and no all red time is assigned in the priorities file.

3.5.4.2 Intersection layout and signal timing data

Information like the intersection layout, NEMA numbering, detailed signal timing plan for each intersection is embedded in “plugin.c”, which is a standard file to make the “dll” file for actuated control. The following is the structure used in the “plugin.c” file. Data structure of “approach_layout” contains detector ID, the number of lanes for the left turn and through movements, left and through movement lane numbers. This information is required to check calls for right of way during red intervals and for gap times during green intervals. Data structure of “approach_layout” is embedded in the data structure of “intersections”. The data structure of “intersections” includes node (intersection in the PARAMICS Zoom window) name and ID, the order of phase sequence, major and minor approaches ID and their directions, the NEMA phase numbering and signal timing plan for each PARAMICS phase. For offset values, the value in the PARAMICS standard input file (priorities file) was used in the API. More detailed description is available in Appendix B.

3.5.5 Control Logic

The objectives of developing an API for coordinated and actuated signal control are two folds. One objective is to develop an API that can consider various phase sequences as shown in Appendix A. It is noted that phase sequences at the traffic signals differ due to characteristics in vehicle arrivals, geometry (turn bay length) and others. The other objective is to develop an API that can control entire intersections in the simulation network. An API for each intersection would be easy to develop but would result in less efficiency.

To maintain a background cycle length and offsets, all coordinated and actuated intersections use the same reference phase – phase for the main street. For an eight-phase controller the reference phase is usually phases ($\phi_2 + \phi_6$) in NEMA phase numbering (10). In order to provide the right of way to the coordinated phase at the yield point within the cycle, the non-synchronized phases are terminated after a certain length of time and these are called force off points. The API calculates all force off points and permissive periods before starting simulation using the “api_setup” control function of the PARAMCIS Programmer.

In the API, coordinated phases (phase $\phi_2 + \phi_6$ in NEMA phase numbering) are set to “maximum recall” (10). These phases utilize their maximum green times plus any unused times from the preceding phases.

All the phases except for coordinated phases could be gapped-out or skipped depending on the demand calls and extension times. This process is established in the “net_action” control function of the Programmer. In order to check demand calls, the “loop_gap” callback function with “APILOOP_INCOMPLETE” option is used in every time step. Depending on the demand call status, the API finds proper path of phase sequence and allocates appropriate interval to the phases on the path using “signal_action” and “signal_inquiry” callback functions.

3.5.6 Fully Actuated Signal Control and T-intersection

The API developed for this study can implement fully actuated signal logic. A significant part of the fully actuated signal logic is similar to that of the actuated-coordinated signal

logic in the priorities file and data structure. However, fully actuated signal control is operated with slightly different control logic. The main difference of fully actuated signal from actuated-coordinated signal is that the fully actuated signal does not need to maintain the background cycle length. The fully actuated signal uses minimum recall on major approaches and no recall on minor approaches. In the fully actuated signal, the dummy phases used in the actuated-coordinated signals are not necessary. Instead, green and red intervals of every phase are operated in a “variable mode”. The green and red interval operated in the “variable mode” can be changed at any time.

T-intersection is a little simpler than 4-leg intersection. The API for T-intersections uses the same input files. However, the control logic inherits most parts of the coordinated actuated signals and fully actuated signals.

3.6 SUMMARY

Building a network for microscopic simulation is not an easy task. The most efficient method of making realistic and acceptable network is to watch the simulation animations. Through visualization, one can identify the locations showing abnormal behavior of individual vehicles and could easily identify the reason for such abnormality.

A procedure of the proposed API for coordinated actuated signal control in PARMICS is presented in this chapter. A few technical challenges encountered during the API development are as follows: i) PARAMICS uses its own fixed phase sequence such that it does not implement NEMA dual-ring type control and ii) yellow change interval was not recalled for the phase once a phase skip occurs. The proposed approach introduced a few

dummy phases to model dual-ring type controller and combined yellow change interval and green times such that only effective green time is utilized. The experience during the development on the API for coordinated actuated signal indicated that the development of an API may not be suitable for practitioners due to complications and efforts required to do so.

CHAPTER 4

BUILDING DYNAMIC O-D ESTIMATION MODELS USING GA AND QUEENSOD MODEL

4.1 INTRODUCTION

As has been mentioned in Chapter 2, several methods including generalized least squares (GLS) model, QUEENSOD model, maximum likelihood estimation (MLE) and Kalman filtering have been used for dynamic O-D estimation.

This study introduces a GA and microscopic traffic simulation based model as a new approach for the dynamic O-D estimation. Even though GA-based models have already been used for O-D estimations (6, 7), those only applied for static O-D estimations on simple networks using a traditional assignment method. However, this study involves a large network and updates assignment matrix (or assignment parameter) using a traffic simulation model (PARAMICS) during the dynamic O-D estimation. The

microscopic traffic simulation model finds the preferred routes and calculates the weights between zone pairs.

One of the practical OD estimation models is selected to produce a benchmark performance such that the performance of the proposed GA-based model is evaluated. QUEENSOD model is selected because it is considered to be the practical OD estimation tool available in the literature. In other words, the QUEENSOD method is simple and is well explained in the literature (12) so that it can be easily implemented in real transportation field. In addition, the QUEENSOD method has been used in a large-scale network consisting of 3,365 nodes, 7,926 links and 565 zones (37).

Both of the GA-based model and the QUEENSOD model find a dynamic O-D matrix which minimizes the discrepancies between estimated and observed link flows. The approach can be mathematically expressed as follow:

$$\min_{\bar{v}_a^t} F(\bar{v}_a^t, v_a^t) \quad (16)$$

subject to

$$\bar{v}_a^t = \sum_{dt} \sum_{ij} \bar{T}_{ij}^{dt} p_{ij,dt}^{a,t}, \quad 0 \leq p_{ij,dt}^{a,t} \leq 1$$

where,

$$F(\bar{v}_a^t, v_a^t) = \text{function of the general error measurement between } v_a^t \text{ and } \bar{v}_a^t,$$

$$v_a^t = \text{observed link flow in link } a \text{ during time interval } t,$$

$$\bar{v}_a^t = \text{estimated link flow in link } a \text{ during time interval } t,$$

$$\bar{T}_{ij}^{dt} = \text{estimated trips leaving zone } i \text{ to zone } j \text{ during time}$$

		interval dt ,
$p_{ij,dt}^{a,t}$	=	proportion of trips leaving zone i to zone j during time interval dt and traveling through link a during time interval t ,
i	=	origin,
j	=	destination,
a	=	link identifier,
dt	=	time interval for departure time, and
t	=	time interval.

The implementations of both GA and QUEENSOD models are presented in this chapter. Firstly, the use of assignment matrix is discussed and followed by the procedure of QUEENSOD method. The selection of GA parameters and settings for the Dynamic O-D estimation model is examined and the procedures are presented.

4.2 USE OF ASSIGNMENT MATRIX

The QUEENSOD model and the proposed GA-based dynamic O-D estimation model use link flows as criteria to match existing traffic conditions. In other words, the models try to reduce the discrepancies between estimates of link flows and observed field link flows. In order to obtain such link volume discrepancies during O-D estimation, an assignment matrix ($p_{ij,dt}^{a,t}$, shown in Equation 16) is generally required. A dynamic and multi-path assignment matrix is selected to calculate the estimates of link flows. This reduces computational time compared to running a microscopic simulation model. However, the assignment matrix is developed from microscopic traffic simulation model. It is also

noted that the microscopic simulation program is used in the evaluation of the O-D matrices developed from both GA and QUEENSOD models.

The assignment matrix includes the origin zone, destination zone, departure time (e.g., 15-minute interval), and time-varying link usage probabilities such that the assignment matrix determines preferred paths between zone pairs and usage probabilities of links on the paths. The assignment matrix is designed to consider the uses of multiple paths and temporal links. Table 3 provides an example of a dynamic and multi-path assignment matrix:

Table 3. Example of an Assignment Matrix

Origin	Destination	Departure Time Interval*	Link ID	Link Usage Probability (by Time)**							
				5:00	5:15	5:30	5:45	6:00	6:15	6:30	6:45
				~ 5:15	~ 5:30	~ 5:45	~ 6:00	~ 6:15	~ 6:30	~ 6:45	~ 7:00
1	2	1	1914	0.75	0.08	0	0	0	0	0	0
1	2	1	1915	0.75	0.08	0	0	0	0	0	0
1	2	1	1918	0.75	0.08	0	0	0	0	0	0
1	2	1	1923	0.75	0.08	0	0	0	0	0	0
1	2	1	1921	0.67	0.17	0	0	0	0	0	0
1	2	1	2007	0.67	0.17	0	0	0	0	0	0
					⋮						
1	2	1	2025	0.58	0.25	0	0	0	0	0	0
1	2	1	2026	0.58	0.25	0	0	0	0	0	0
1	2	1	1914	0.17	0	0	0	0	0	0	0
1	2	1	1915	0.17	0	0	0	0	0	0	0
1	2	1	1918	0.17	0	0	0	0	0	0	0
1	2	1	1923	0.17	0	0	0	0	0	0	0
1	2	1	1921	0.17	0	0	0	0	0	0	0
1	2	1	2007	0.17	0	0	0	0	0	0	0
					⋮						
1	2	1	3100	0.08	0.08	0	0	0	0	0	0
1	2	1	2032	0.08	0.08	0	0	0	0	0	0

* Departure time of 1 means 5:00 pm ~ 5:15 pm.

** All floating values are rounded from the third decimal point.

As can be seen in Table 3, vehicles traveling from zone 1 (origin) to zone 2 (destination) during time interval 1 (5:00 pm ~ 5:15 pm) use two paths:

1) Path 1: 1914→1915→1918→1923→1921→2007→ → 2025 → 2026

2) Path 2: 1914→1915→1918→1923→1921→2007→ → 3100 → 2032

It can be also seen from Table 3 that 83% of the vehicles use path 1 and 17% use path 2.

These percentages represent the sums of link usage probabilities for each link in Path 1 (0.75 + 0.08 or 0.67 + 0.17 or 0.58 + 0.25) and 2 (0.17 or 0.08 + 0.08). It can also be observed that among the 83% of the vehicles following path 1, 75% use link 1914 during the period 5:00 pm ~ 5:15 pm and 8% use the link during the period 5:15 pm ~ 5:30 pm. In addition 17% of vehicles following the path 2 also use link 1914 during the period 5:00 pm ~ 5:15 pm.

4.3 QUEENSOD MODEL

4.3.1 Mechanism of QUEENSOD Model

QUEENSOD, introduced by Van Aerde *et al.* (12), is a model for generating static and dynamic synthetic O-D matrices. The QUEENSOD model initiates the first iteration from a seed O-D matrix. The seed O-D can be either a uniform or historic O-D matrix. The seed O-D matrix is utilized to generate estimates of link flows based on the estimates of drivers' expected route choices. The seed O-D matrix is adjusted on the basis of the quantitative comparisons between observed and estimated link flows. Through the multiple iterations of these processes, the seed O-D matrix is systematically modified to

produce a new and better O-D matrix. The following procedure explains the process of QUEENSOD model:

- 1) Step 1: Determining dynamic drivers' expected route choices (a dynamic and multi-paths assignment matrix)
- 2) Step 2: Determining an appropriate dynamic seed O-D matrix (uniform O-D matrix or historic O-D matrix)
- 3) Step 3: Conducting assignment using dynamic seed O-D matrix (assignment of dynamic seed O-D matrix to the network based on the assignment matrix)
- 4) Step 4: Estimating link error correction factors

$$CL_a^t = \begin{cases} \frac{V_a^t}{\hat{V}_a^t} & \text{if } \hat{V}_a^t > 0 \\ 1.0 & \text{if } \hat{V}_a^t = 0 \end{cases} \quad (17)$$

where,

- CL_a^t = correction factor for link a and time interval t ,
- V_a^t = actual link volume at link a and time interval t ,
- \hat{V}_a^t = estimated link volume at link a and time interval t ,
- a = link identifier, and
- t = time interval.

- 5) Step 5: Estimating O-D error correction factors

$$\begin{aligned} CS_{ij}^{dt} &= \prod_{a,t} (PR_{ij,a}^{dt,t} \times CL_a^t) \\ PO_{ij}^{dt} &= \sum_t \sum_a PR_{ij,a}^{dt,t} \\ CF_{ij}^{dt} &= \begin{cases} CS_{ij}^{dt} (PO_{ij}^{dt})^{-1} & \text{if } PO_{ij}^{dt} > 0 \\ 1.0 & \text{if } PO_{ij}^{dt} = 0 \end{cases} \end{aligned} \quad (18)$$

where,

- CS_{ij}^{dt} = correction factors summed between origin i and destination j ,
- $PR_{ij,a}^{dt,t}$ = probability that vehicles leaving i to j during dt use link a during time interval t ,
- PO_{ij}^{dt} = probabilities of link use summed for all links used for traveling from i to j during time interval t ,
- CF_{ij}^{dt} = correction factors finalized for between origin i and destination j at time interval t ,
- i = origin,
- j = destination, and
- dt = departure time interval.

6) Step 6: Estimating a new O-D matrix

$$OD_{ij}^{dt}(n+1) = OD_{ij}^{dt}(n) \times CF_{ij}^{dt} \quad (19)$$

where,

- $OD_{ij}^{dt}(n+1)$ = dynamic O-D matrix at iteration $n+1$, and
- n = iteration identifier.

7) Step 7: Repeating above steps until convergence criterion is met

4.3.2 Using Existing Traffic Information

The traffic simulation network built in Chapter 3 (Figure 11) has seven external zones that connect the City of Hampton to other areas. Data of 15-minute interval flows are available for the links connected to the external zones. Thus, it is assumed that the 15-minute interval link flows at external zones represent the total of O-D demands generated from the external zones during each of 15-minute departure time interval.

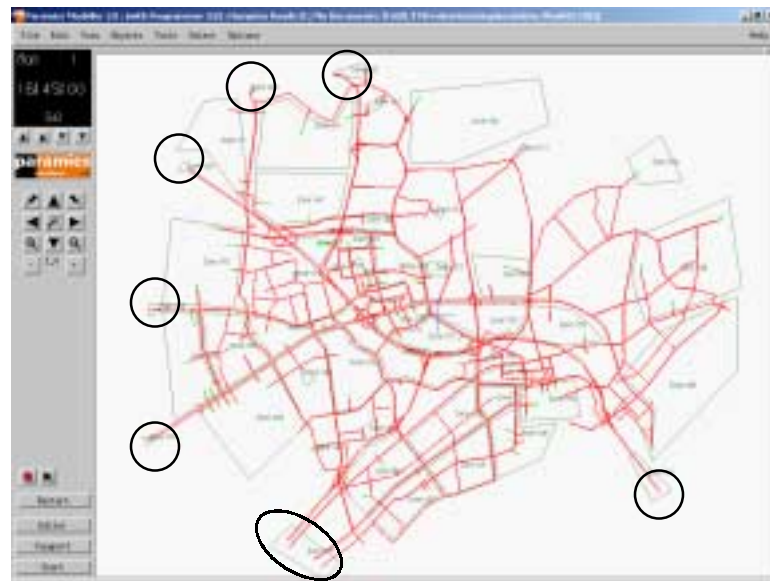


Figure 11. Seven External Zones of Case Study Network

Thus, the estimates of dynamic O-D demands generated from external zones are fixed by matching to the counted field link volumes (15-minute interval) after estimating a new O-D matrix (step 6 in Section [4.3.1]) in every iteration.

4.3.3 QUEENSOD-based O-D Estimation Process

The QUEENSOD model is evaluated according to the following procedure:

(Step #1): Finding better assignment matrix

- (1) Determine uniform O-D matrix as a seed O-D matrix
- (2) Make assignment matrix based on the uniform O-D using simulation model
- (3) Run QUEENSOD model

(4) Evaluate iterations using simulation model

(5) Find the best O-D pattern

(Step #2) Finding the best O-D matrix

(1) Making a new assignment matrix from the selected O-D in Step #1

(2) Run QUEENSOD model

(3) Evaluate iterations using simulation model

(4) Find the best O-D pattern

However, if any historical O-D matrix were available, the first step of QUEENSOD-based dynamic O-D estimation process could be skipped because the assignment matrix can be directly calculated from the historical O-D matrix. In addition, the historical O-D matrix can be used as a seed O-D matrix for QUEENSOD-based dynamic O-D estimation.

4.4 GA-BASED O-D ESTIMATION MODEL

As explained in Chapter 2, the GA model has a capability of finding a good solution within a reasonable time and works reasonably well with a complicated large scale optimization. The proposed GA-based O-D estimation model works with a population that is consisted of a large number of randomly generated potential O-D matrices.

Through the GA selection based on the extent of discrepancy between actual observed link flows and estimated link flows calculated from the product of the O-D matrix and assignment matrix, better O-D matrices are selected for crossover and mutation

operations. According to schema theorem (add reference), the number of better O-D matrices (i.e., the O-D matrices with less discrepancies between observed link flows and estimated link flows) increases through generations.

The GA-based O-D estimation model is designed to overcome some problems identified in the QUEENSOD-based model, which will be explained in detail later in this chapter and Chapter 5. For example, the GA model generates O-D matrices for a given total O-D demand. Note that the QUEENSOD model could not constrain total O-D demand volume during OD estimation. This section describes the process of estimating an appropriate total O-D demand and an O-D matrix, and it also explains the procedure of developing the proposed GA-based model. The reason for using an assignment matrix instead of a simulation model in the estimation of link volumes during GA generations is explained at the beginning of this section. This section also provides the selection of solution representation, parameters and settings for the GA-based OD estimation model.

4.4.1 Determination of GA Selection Method, GA Parameters and GA Operators

4.4.1.1 GA Selection Method

Selection of individuals (or parents) plays an extremely important role in GA. The selection of individuals in GA is based on the individual's fitness such that better individuals have higher probability of being selected. An individual in the population can be selected more than once.

This study compared two selection methods, roulette wheel selection and normalized geometric ranking method (24), using a binary GA with a population size of 200, a maximum number of generation of 50, a simple crossover, and a simple mutation.

The comparison between roulette wheel selection and normalized geometric ranking method showed that normalized geometric ranking selection performed better. Figure 12 shows the selection probabilities of randomly generated 200 solutions for dynamic O-D estimation using Equation (10) (roulette wheel selection) and Equation (11) (normalized geometric ranking selection). The x-axis gives the ranking of chromosomes from 1 to 200 (chromosome of rank 1 is better than chromosome of rank 2 and so on) and the y-axis the selection probability. It can be observed that the selection probabilities of the better chromosomes under ranking method are significantly higher than those of roulette wheel selection.

In the normalized geometric ranking method better solutions (i.e., those with smaller error measure values) are given higher rankings, while in the roulette wheel method better solutions are given higher probabilities according to the proportions. One of the shortcomings in the roulette wheel method is that the selection probabilities of individuals become not much distinguishable when fitness values of individuals are similar. However, the ranking method selects individuals based on the rankings such that it can always select better solutions regardless of the characteristics fitness values as shown Figure 12. The elitist method is combined with the normalized geometric selection method in order to keep the best solution over generations.

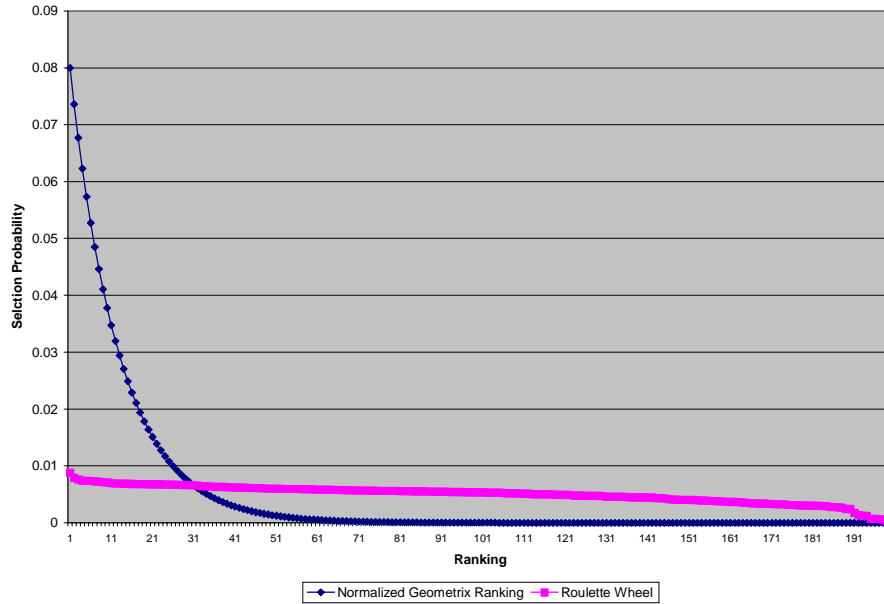


Figure 12. Selection Probability Comparison of Two GA Selection Methods

4.4.1.2 GA Operators

This section examines two types of GA models (binary and real-valued GA). A total of nine combinations of operators: three cross over operator (arithmetic crossover, heuristic crossover, and simple crossover) and three mutations (multi non-uniform mutation, non-uniform mutation and uniform mutation) were tested for the real-valued GA. Among these, simple crossover and multi non-uniform mutation combination is selected as it provided better results than other combinations. However, the results of other combinations were very similar to the best combination. For the binary GA model, simple mutation with a probability of 0.05 and simple crossover with a probability of 0.5 are selected.

4.4.1.3 Number of Generation and Stop condition

The most frequently used stopping criterion of the GA model is to have a specific maximum number of generations (24). A maximum number of generation of 50 was selected as the improvement in the best solution was found to be insignificant after 50 generations. Two other stopping criteria were used in conjunction with the maximum number of generations: 1) lack of improvement in the fitness of the best solution over 10 generations, 2) no differences between the fitness of best solution and the average fitness of the entire solutions over 10 generations.

$$\frac{Best_Fitness^{t-5} - Best_Fitness^t}{Best_Fitness^{t-5}} \leq 0.00001$$

and

$$\frac{Mean_Fitness^t - Best_Fitness^t}{Best_Fitness^t} \leq 0.00001 \quad (20)$$

where,

$Best_Fitness^t$ = fitness of the best solution at generation t ,
 $Mean_Fitness^t$ = average fitness of entire solutions at generation t , and
 t = generation number, $t > 10$.

4.4.2 Selection of Solution Representation, GA model and Evaluation Function

In GA, each individual solution is systemically represented by a chromosome-like data structure. The representation scheme determines how the problem is structured in GA. Evaluation function in GA generates the fitness values of individuals in a population.

Based on the fitness values, GA conducts a selection of “parents” for GA operations. Thus, GA representation scheme and evaluation function have to be well designed. A total of 12 combinations including three proposed solution representation schemes, two GA models (binary and real-valued GA) and two evaluation functions are tested in this section.

The selection of solution representation and GA model (binary or real-valued GA) is explained and followed by the selection of evaluation function.

4.4.2.1 Solution Representation Scheme

More efficient and natural solution representation schemes produce faster convergence and better solutions (25). Major focuses on the design of solution representation in this study are twofold: i) the utilization of the existing flow patterns for external zones and ii) natural representation of dynamic O-D matrix.

In order to utilize existing flow patterns, as mentioned in the QUEENSOD model (refer to Section [4.3.2]), the proposed solution representations use 15-minute interval link flows. The natural representation is related to the structure of genes (or parameters) in the solution. The proposed GA-based dynamic OD estimation method is designed to maintain the total O-D demand (i.e., fixed during optimization) as an exogenously given value such that the total O-D demand does not change over GA generations. This ensures GA convergence. It is noted that total O-D demand in the QUEENSOD model could not be constrained during iterations. Thus, the increase in the total O-D demand might result in unrealistic values after a large number of iterations. In order to maintain the total O-D

demand, the proposed solution representations utilize proportional allocation approaches.

The following three solution representation schemes are proposed:

- 1) Solution representation 1: proportional allocation approach without utilizing link flows connected to external zones,
- 2) Solution representation 2: proportional allocation approach utilizing 1-hour interval link flows connected to external zones, and
- 3) Solution representation 3: proportional allocation approach utilizing 15-minute interval link flows connected to external zones.

Solution Representation 1

This solution representation simply uses the proportional allocation approach through three sequent stages (i.e., the parameters of one solution representation can be grouped into three sequential parts and the parameters in each group have different roles in the dynamic O-D estimation) to make an O-D matrix. The four parameters in the first group are used to divide a given total O-D demand into four O-D demand pairs that depart in each of four 15-minute time intervals. Based on the four O-D demand pairs, the parameters in the second group make sums of O-D demands that depart in 15-minute intervals from each zone. Finally, the parameters in the third group determine the values of the O-D demands that depart from every origin to every destination. One solution (or chromosome) consists of 10,004 parameters (or genes). The detailed decoding equation is expressed in Appendix C-1.

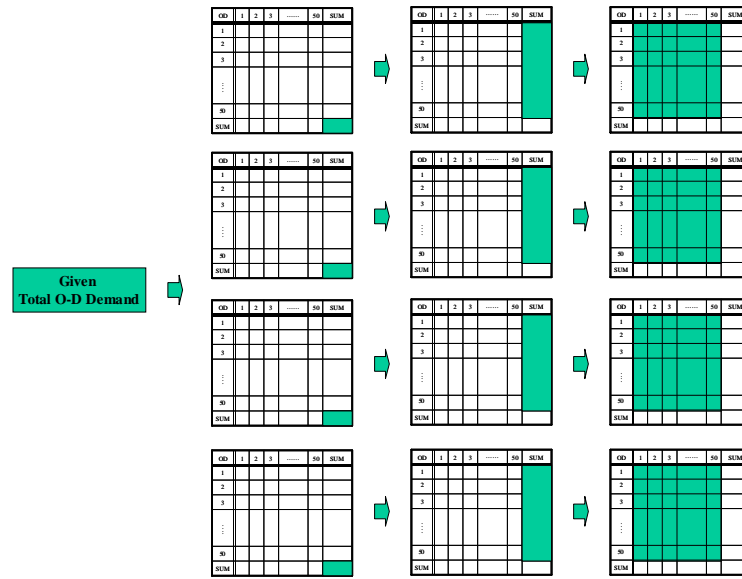


Figure 13. Three Stages of Making O-D Matrix Using Solution Representation 1

Solution Representation 2

This solution representation uses proportional allocation approach that utilizes 1-hour interval link flows connected to external zones through three sequent stages. The parameters in the first group are used to divide a given total O-D demand into four O-D demand pairs that depart from internal zones during a 1-hour period. This differs from solution representation 1 in that only the sums of O-D demands starting from internal zones are determined in this stage. In the case of the seven external zones, the counted link flows for a 1-hour interval are directly used instead of being calculated. Based on the sums of O-D demands, parameters in the second group make O-D demands that depart from/to every zone for a 1-hour interval. Finally, parameters in the third group divide these 1-hour O-D demands into 15- minute O-D demands from each origin to every destination for every 15-minute time interval. One solution consists of 9,843 parameters. The detailed decoding equation is expressed in Appendix C-2.

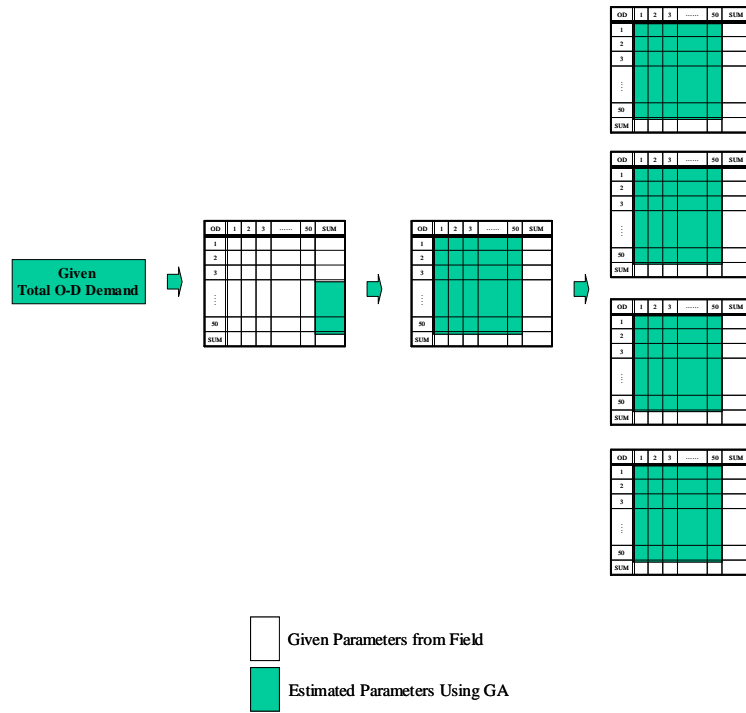


Figure 14. Three Stages of Making O-D Matrix Using Solution Representation 2

Solution Representation 3

This solution representation uses the proportional allocation approach utilizing 15-minute interval link flows through three sequent stages. Parameters in the three groups are used in the same way as solution representation 1. However, the first and second stages apply for O-D demands generated from only internal zones. In the case of external zones, the counted link flows for 15-minute intervals are directly used. One solution consists of 9,976 parameters. The detailed decoding equation is expressed in Appendix C-3.

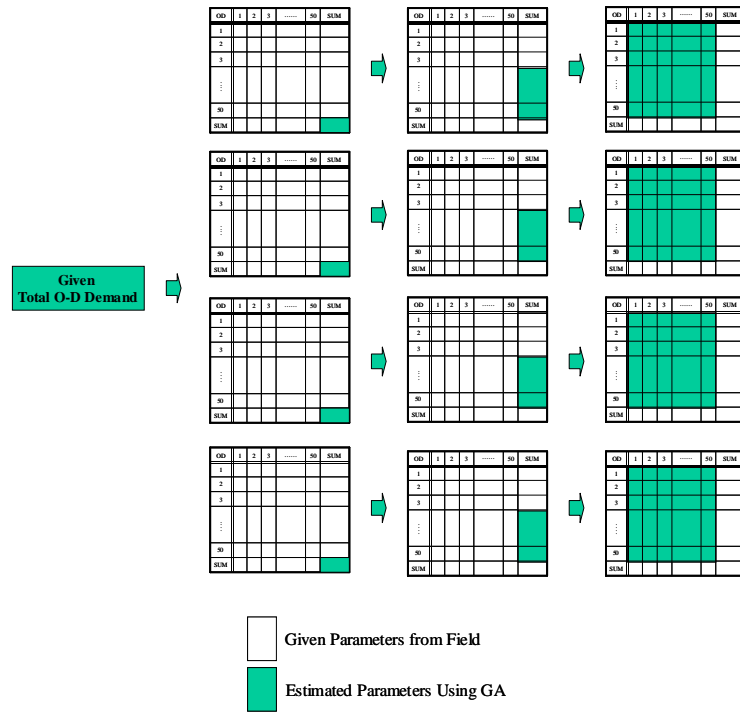


Figure 15. Three Stages of Making O-D Matrix Using Solution Representation 3

4.4.2.2 Binary and Real-valued GA Models

In Holland's original design, the solution representation was limited to binary digits.

However, a real-valued GA is later proposed because it can handle the problem closer to the actual problem representation that offers higher precision with more consistent results across replications (22). The real-valued GA is more efficient in terms of CPU time and computer memory usage. The above three solution representations using binary and real-valued GA models are tested in the next section.

4.4.2.3 Comparison Performance of Solution representations and GA models

In order to select a more efficient GA model and its solution representation, this study examines the performances of three solution representations using both binary and real-

valued GAs. For the purpose of evaluating the GA performance across replications, a maximum generation number of 25, initial population size of 50, and total O-D demand of 45,000 are used. In order to ensure unbiased comparisons, the same initial population was used for all the cases. The GA selection method, GA operators and parameters are previously described.

Capability of Generating Initial Populations

In order to test the performance of the three solution representations in making initial population, three populations sets with 50 individual solutions each are randomly generated using the three solution representation schemes. As can be observed in Table 4, the solution representations 2 and 3 that utilized link flows at external zones show better results. Solution representation 2 is found to be better than solution representation 3.

Table 4. Summary of Initial Populations From Three Solution Representations

Solution Representation	Error Measure	Mean	Median	Standard Deviation	Minimum	Maximum
1	SSE	4.72E+07	4.63E+07	3.22E+06	4.22E+07	5.68E+07
	MAPE	69.04332	69.4485	3.116729	63.629	75.98
2	SSE	3.92E+07	3.86E+07	2.46E+06	3.60E+07	4.70E+07
	MAPE	59.85404	59.7125	1.969855	55.555	64.458
3	SSE	4.33E+07	4.24E+07	2.95E+06	3.86E+07	5.21E+07
	MAPE	63.0396	63.4095	2.845755	58.096	69.373

Capability of GA Representation

As shown in Table 5, the binary GA with solution representation 2 produced the best performance in GA representation. Based on this result, the binary GA with solution representation 2 is used for dynamic O-D estimations.

Table 5. Summary of Examination

Solution Representations	Statistics	GA Models			
		Binary GA		Real-Valued GA	
		MAPE Eval. Func.	SSE Eval. Func.	MAPE Eval. Func.	SSE Eval. Func.
1	Best	55.40	3.49E+07	56.12	3.64E+07
	Mean	58.55	3.56E+07	58.59	3.73E+07
	STD	1.44	1.12E+06	1.28	1.09E+06
2	Best	52.61	3.21E+07	53.75	3.47E+07
	Mean	54.47	3.38E+07	55.42	3.62E+07
	STD	0.81	7.55E+05	1.36	8.12E+05
3	Best	53.45	3.43E+07	55.97	3.68E+07
	Mean	55.70	3.61E+05	57.62	3.85E+07
	STD	0.96	1.02E+05	1.46	1.27E+05

4.4.3 Evaluation Function

4.4.3.1 Evaluation Process

Evaluation function in GA generates fitness values of individuals in the population.

Based on the fitness values, GA conducts a selection of “parents” for GA operations in the next process. The evaluation function involves three sequential processes: 1)

Calculation of O-D matrix using individual solution generated by GA, 2) Estimation of

link volumes using calculated O-D matrix and dynamic and multi-path assignment matrix, and 3) Calculation of fitness values based on the error measure.

4.4.3.2 Measures of Effectiveness (MOE) and Error Measure

In the calculation of MOE, the 15-minute interval link flows are used. To estimate the link volumes, the evaluation function uses the dynamic and multi-path assignment matrix generated from the PARAMICS simulation under initial O-D (or uniform O-D) demand. The evaluation functions considered in GA are the Sum of Squared Error (SSE) and Mean Absolute Percent Error (MAPE). The SSE and MAPE measures are obtained from actual observed link flows and estimated link flows from the assignment matrix and estimated OD matrix. The SSE uses a squared error as shown in Equation (7), whereas the MAPE is based on an averaged relative error (i.e., the absolute deviation between estimated link volume and actual link volume are divided by actual link volume) and is described in Equation (8).

- Sum of Squared Error (SSE)

$$\sum_{a=1}^N \sum_{t=1}^{n_t} (v_a^t - \bar{v}_a^t)^2 \quad (7)$$

- Mean Absolute Percent Error (MAPE)

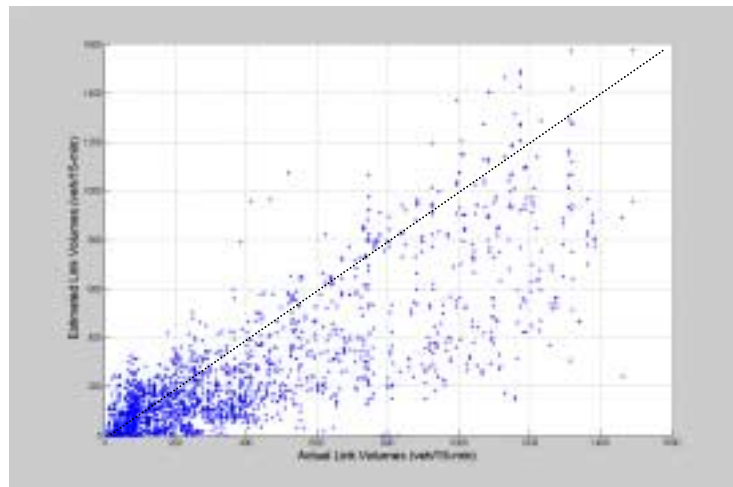
$$\frac{\sum_{a=1}^N \sum_{t=1}^{n_t} \left(\frac{|\bar{v}_a^t - v_a^t|}{v_a^t} \right)}{N} \times 100 \quad (8)$$

where,

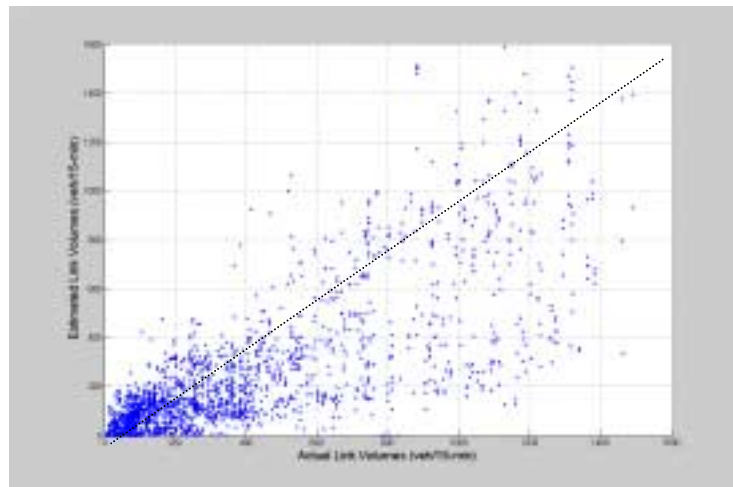
a = link identifier,

\bar{v}_a^t	=	estimated flow on link a at time interval t ,
v_a^t	=	observed flow on link a at time interval t ,
t	=	time interval,
n_t	=	number of time interval, and
N	=	number of links.

The GA under the SSE evaluation function tries to find solutions with low deviations of link volumes on major roads, because the squared deviations between estimated and actual link volumes on major roads are bigger than those of minor roads. However, the MAPE is a unit free error measure as it averages relative deviations (35). Hence, the MAPE considers the volume deviations on major and minor roads with the same weight. The following Figure 16 shows the results of GA under the MAPE and the SSE evaluation functions.



(a) GA Using SSE Evaluation Function



(b) GA Using MAPE Evaluation Function

Figure 16. Estimated and Observed Link Flows Comparison

As mentioned in the above, the GA with SSE seems to overestimate the low volume links and yet to perform better with high volume links. This situation is apparent in Figures 16 and 17, and Table 6.

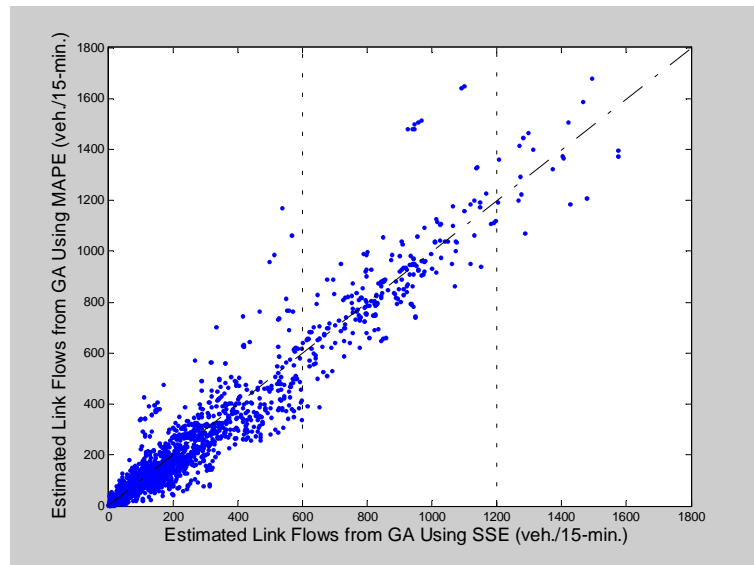


Figure 17. Correlation of Estimated Link Flows from GA Using SSE and MAPE

In order to investigate the performance of each error measure, the links were divided into three groups. The first group includes links with volumes less than 600 veh/15-min. The second group includes links with volumes between 600 veh/15-min and 1,200 veh/15-min. The remaining links are categorized as the third group. The SSE and MAPE of the link volume estimates from GA are then compared by groups. As can be observed in Table 6, GA with SSE shows better results in groups 2 and 3.

Table 6. Error Measures in Divided Groups

Evaluation Function	Link Volume < 600 (Unit: veh/15-min)		600≤Link Volume < 1200 (Unit: veh/15-min)		Link Volume > 1200 (Unit: veh/15-min)	
	SSE	MAPE	SSE	MAPE	SSE	MAPE
SSE	3.73E+06	65.42%	6.70E+06	38.02%	2.20E+07	38.03%
MAPE	3.70E+07	56.08%	80233645	40.93%	2.59E+07	40.23%

Though MAPE, a unit free measure (35), is widely used, it does not distinguish heavy and light traffic volume conditions. Since major roads having higher volumes are more important than minor roads in the O-D estimation and of course other traffic engineering studies, the SSE has been selected as an error measure for GA evaluation function. Finally, the GA model mathematically follows Equation (16).

$$\min_{\bar{v}_a^t} \sum_{a=1}^N \sum_{t=1}^{n_t} (v_a^t - \bar{v}_a^t)^2 \quad (16)$$

subject to

$$\bar{v}_a^t = \sum_{dt} \sum_{ij} \bar{T}_{ij}^{dt} p_{ij,dt}^{a,t}, \quad 0 \leq p_{ij,dt}^{a,t} \leq 1$$

where,

v_a^t	=	observed link flow in link a during time interval t ,
\bar{v}_a^t	=	estimated link flow in link a during time interval t ,
\bar{T}_{ij}^{dt}	=	estimated trips leaving zone i to zone j during time interval dt ,
$p_{ij,dt}^{a,t}$	=	proportion of trips leaving zone i to zone j during time interval dt and traveling through link a during time interval t ,
i	=	origin,
j	=	destination,
a	=	link identifier,
dt	=	time interval for departure time, and
t	=	time interval.

4.4.4 GA-based O-D Estimation Process

The proposed GA-based dynamic O-D estimation involves the following steps:

(Step #1): Finding better assignment matrix

- (1) Determine uniform O-D matrix as a seed O-D matrix
- (2) Make assignment matrix based on the uniform O-D using simulation model
- (3) Run GA with different sums of O-D demands
- (4) Evaluate GA runs
- (5) Find the best O-D pattern

(Step #2) Finding appropriate sum of O-D demands

- (1) Making a new assignment matrix from the selected O-D using simulation model
- (2) Run GA with different sums of O-D demands
- (3) Evaluate GA runs

(4) Find the appropriate sum of O-D demands

(Step #3) Finding the best O-D matrix

(1) Making a new assignment matrix from the selected O-D using simulation model

(2) Run GA with different sums of O-D demands near to the best one in Step #2

(3) Evaluate GA runs with

(4) Find the appropriate sum of O-D demands

Because information about target O-D demand or historical O-D matrix is assumed to be unknown, this study starts with a uniform O-D matrix to make an assignment matrix. However, starting from the uniform O-D matrix is neither realistic nor acceptable to develop a realistic assignment matrix. Therefore, the main O-D estimation begins from step two. In the step two, runs of GAs with different total O-D demand that covers the possible range of true total O-D demand are conducted. The results with multiple traffic simulation runs are then evaluated to select the appropriate total O-D demand. In the third step, a new assignment matrix is calculated using the results from step two. Using the new assignment matrix and several total O-D demands that are close to the best total O-D demand in step two, the step three finds the best O-D matrix. The assignment matrix is improved over three steps. However, if any historical O-D matrix were available, the first step of GA-based dynamic O-D estimation process could be skipped because the assignment matrix can be directly calculated from the historical O-D matrix.

4.5 SUMMARY

The QUEENSOD model is a simple and straightforward and well explained in the work of Van Aerde et al. (12). Thus, this report followed the approach proposed by Van Aerde. However, the GA model is used for only static O-D estimations (6, 7). Thus, this study investigates the performance of GA models with different methodology and parameters (i.e., binary GA vs. real-valued GA). After the extensive evaluation of GA settings and parameters, the binary GA model, with the simple crossover, the simple mutation, the normalized geometric selection, the SSE in the evaluation function and solution representation 2, was selected.

In addition, the GA-based dynamic O-D estimation model can also accommodate the historical O-D matrix with changes in the object function. In other words, the object functions can be extended from a least square formula based on the bilevel program approach proposed by Yang (28) [refer to Equation (14)].

The proposed GA-based dynamic OD estimation method is designed to maintain the total O-D demand (i.e., fixed during optimization) as an exogenously given value such that the total does not change during GA generations. This ensures GA convergence. On the contrary, total O-D demand in the QUEENSOD model is not constrained during its iterations.

CHAPTER 5

IMPLEMENTATION OF DYNAMIC O-D MATRIX ESTIMATION MODELS

5.1 INTRODUCTION

In this chapter, the proposed GA-based dynamic O-D estimation and the QUEENSOD-based dynamic O-D estimation are implemented with the City of Hampton network built in with the PARAMICS microscopic simulation program. Both dynamic O-D estimation models are coded using MATLAB 6.5 (36).

5.1.1 Data Collection

Link counts for the dynamic O-D estimation were available from the City of Hampton and the VDOT Database. The database, which operated by the Virginia department of transportation (VDOT), provides traffic information (i.e., link counts) of continuous count stations and non-continuous count stations in Virginia. Most of the link counts

were collected in 2001 with 15-minute or 1-hour intervals. It is noted that these link counts data did not cover entire network and they were not collected on a same day.

5.1.2 Peak Period Selection

Since most of the traffic engineering studies and ITS applications intend to improve the capacity and efficiency of traffic facilities during peak period, this study considers a simulation-based test-bed with a dynamic O-D during the peak hour. In order to identify the peak period in the City of Hampton, 656 directional link volumes between 6 am and 7 pm were converted into 15-minute interval link flows. It is observed that the time period between 5 pm and 6 pm is the peak hour. Figure 18 shows the temporal pattern of 15-minute interval link flows of randomly selected five links on Mercury Blvd.

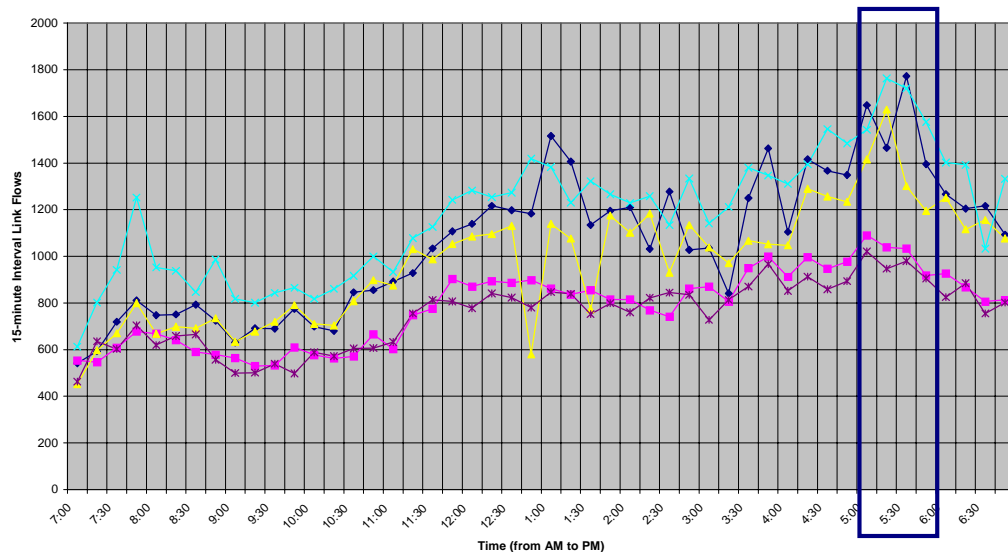


Figure 18. 15-minute Interval Link Flows

5.2 QUEENSOD-BASED DYNAMIC O-D ESTIMATION

In the QUEENSOD model, the maximum iteration of 50 is used. The estimates of the dynamic O-D matrix are summarized with the SSE obtained from observed actual link flows and estimated link flows by 15-minute interval. This is because individual 15-minute interval estimates of the dynamic O-D matrix follow different trends as shown in Figures 19 and 22.

5.2.1 Finding Better Assignment Matrix (Step #1)

In the first step (i.e., initialization period), a uniform seed O-D matrix is used to create an assignment matrix in which obtained from the PARAMICS simulation run. The results of the QUEENSOD model estimation are illustrated in Figure 19 and summarized in Table 7. In Figure 19, it can be observed that the average SSE values dramatically decrease during the first two iterations and gradually decrease up to 25 iterations.

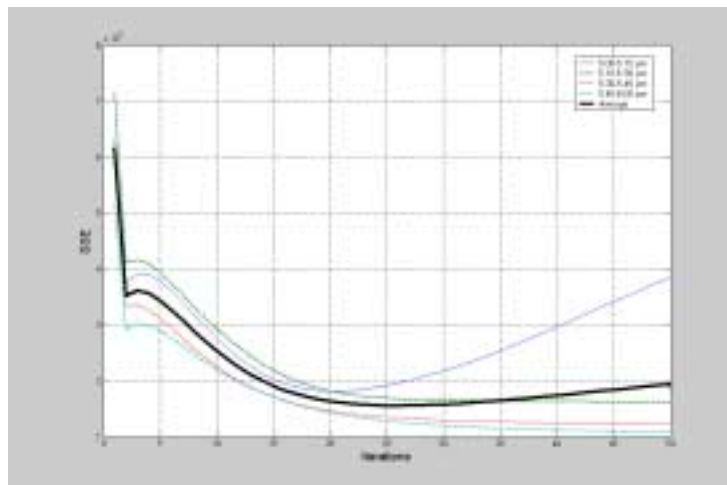


Figure 19. Convergence of the QUEENSOD Model in Step 1

Figure 20 also shows the total O-D demand over the iterations of the QUEENSOD estimation. Note that in the QUEENSOD model, the adjustment of each O-D pair demand is independent from other O-D pairs such that the total O-D demand is not constrained during optimization. Thus, the total O-D demand changes over iterations. It was found that after 3 or 4 iterations, the QUEENSOD method kept increasing total O-D demand and that resulted in lower SSE values. Further investigation revealed that the QUEENSOD method increased O-D demands for those O-D pairs having higher SSE errors. This was because the initial O-D demands were much lower than “unknown optimal” O-D demands.

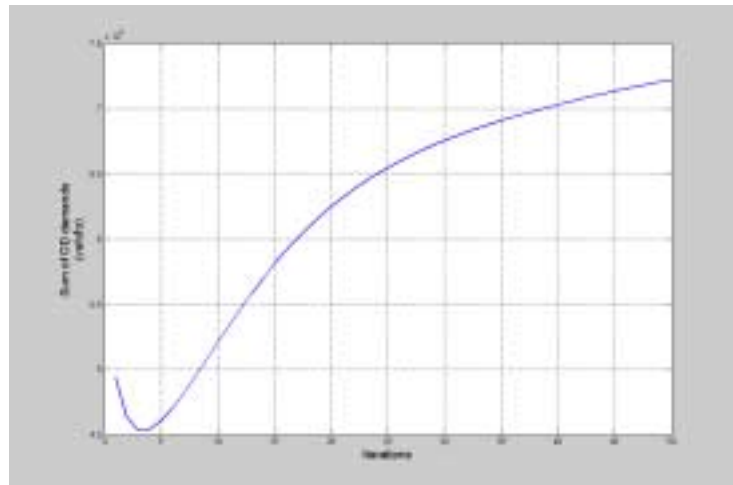


Figure 20. Change of Total of O-D Demand in Iterations

The estimated O-D matrices from the QUEENSOD model are then evaluated using the PARAMICS simulation program. Each O-D matrix is simulated for five times with different random number seeds to consider stochastic variability. It is noted that the PARAMICS program simulates on the basis of dynamic traffic assignment and provides

dynamic link traffic counts and assignment matrix. The results of O-D matrices evaluation in PARAMICS are presented in Figure 21 using a box-plot graph and Table 8. Note that only the first nine O-D matrices results are presented as O-D matrices after 10th iterations revealed extreme congestions during the PARAMICS simulations. The O-D matrix obtained at the 8th iteration produced the best result.

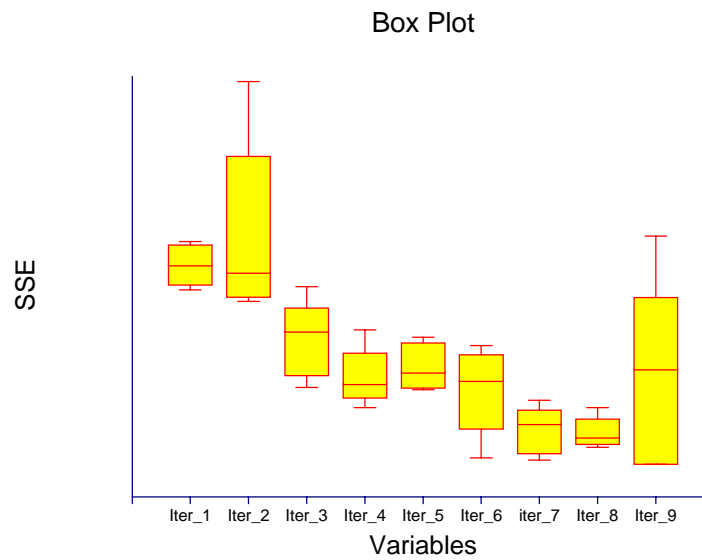


Figure 21. Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS

5.2.2 Finding the Best O-D Matrix (Step #2)

In step 2, a new assignment matrix was generated from the PARAMICS simulation using the best O-D matrix obtained at the 8th iteration during step 1. The QUEENSOD method is implemented with the new assignment matrix and a new seed O-D matrix (i.e., the best O-D matrix obtained at the 8th iteration during step 1). The QUEENSOD convergence is plotted in Figure 22 and summarized in Table 7. Interestingly, the SSE increased for the

first 5 iterations and then decreased. However, the changes in the total O-D demand, shown in Figure 23, presented similar pattern when compared to that of step 1.

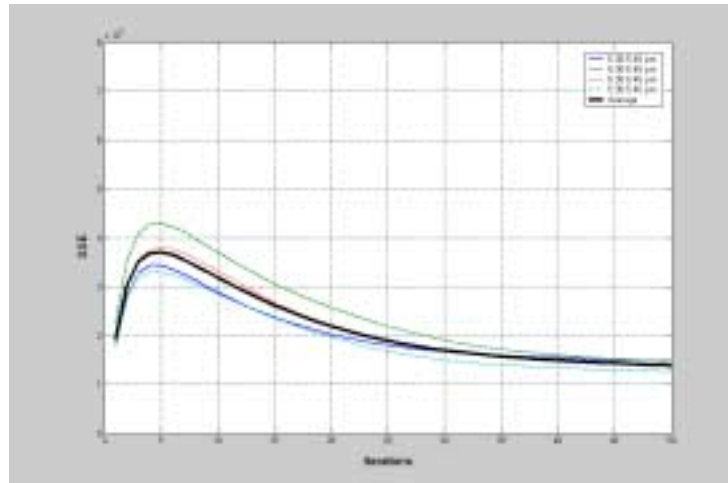


Figure 22. Convergence of the QUEENSOD Model In Step 2

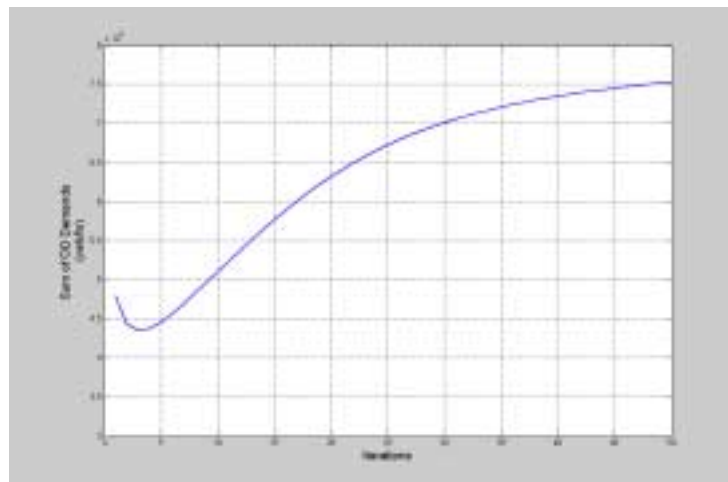


Figure 23. Change of Total O-D Demand in Iterations

Table 7. Results of the QUEENSOD Iterations

Iteration	SSE		Iteration	SSE	
	Step 1	Step 2		Step 1	Step 2
1	6.17E+07	1.93E+07	26	1.55E+07	1.85E+07
2	3.52E+07	3.02E+07	27	1.56E+07	1.81E+07
3	3.62E+07	3.52E+07	28	1.56E+07	1.77E+07
4	3.56E+07	3.69E+07	29	1.57E+07	1.73E+07
5	3.43E+07	3.71E+07	30	1.58E+07	1.70E+07
6	3.26E+07	3.65E+07	31	1.59E+07	1.67E+07
7	3.08E+07	3.55E+07	32	1.60E+07	1.64E+07
8	2.89E+07	3.44E+07	33	1.61E+07	1.62E+07
9	2.70E+07	3.32E+07	34	1.63E+07	1.60E+07
10	2.53E+07	3.19E+07	35	1.64E+07	1.58E+07
11	2.38E+07	3.07E+07	36	1.66E+07	1.56E+07
12	2.23E+07	2.95E+07	37	1.68E+07	1.54E+07
13	2.11E+07	2.84E+07	38	1.70E+07	1.53E+07
14	2.00E+07	2.73E+07	39	1.71E+07	1.51E+07
15	1.91E+07	2.63E+07	40	1.73E+07	1.50E+07
16	1.83E+07	2.53E+07	41	1.75E+07	1.48E+07
17	1.76E+07	2.44E+07	42	1.77E+07	1.47E+07
18	1.71E+07	2.35E+07	43	1.80E+07	1.46E+07
19	1.66E+07	2.27E+07	44	1.82E+07	1.45E+07
20	1.63E+07	2.19E+07	45	1.84E+07	1.44E+07
21	1.60E+07	2.12E+07	46	1.86E+07	1.43E+07
22	1.58E+07	2.06E+07	47	1.88E+07	1.42E+07
23	1.57E+07	2.00E+07	48	1.90E+07	1.41E+07
24	1.56E+07	1.95E+07	49	1.92E+07	1.40E+07
25	1.55E+07	1.90E+07	50	1.94E+07	1.39E+07

Again, the PARAMIC simulation results of the first nine O-D matrices are shown in Table 8 and Figure 24. It was found that the O-D matrix at 8th iteration resulted in the best estimation.

Table 8. Evaluation Results of Estimated O-D Matrix from PARAMICS

Iteration Num.	Statistic	SSE	
		Step 1	Step 2
1	Average	4.24E+07	6.21E+07
	Median	4.10E+07	6.21E+07
	STD	3.32E+06	1.94E+06
2	Average	4.27E+07	6.48E+07
	Median	4.26E+07	6.14E+07
	STD	7.35E+05	8.61E+06
3	Average	4.36E+07	5.50E+07
	Median	4.29E+07	5.58E+07
	STD	2.77E+06	3.66E+06
4	Average	4.43E+07	5.14E+07
	Median	4.37E+07	5.08E+07
	STD	1.48E+06	2.76E+06
5	Average	4.51E+07	5.24E+07
	Median	4.55E+07	5.19E+07
	STD	2.64E+06	2.18E+06
6	Average	4.42E+07	5.02E+07
	Median	4.50E+07	5.11E+07
	STD	1.95E+06	4.12E+06
7	Average	4.33E+07	4.63E+07
	Median	4.34E+07	4.70E+07
	STD	1.12E+06	2.25E+06
8	Average	4.14E+07	4.61E+07
	Median	4.13E+07	4.57E+07
	STD	1.33E+06	1.46E+06
9	Average	4.18E+07	5.13E+07
	Median	4.16E+07	5.22E+07
	STD	1.07E+06	8.97E+06

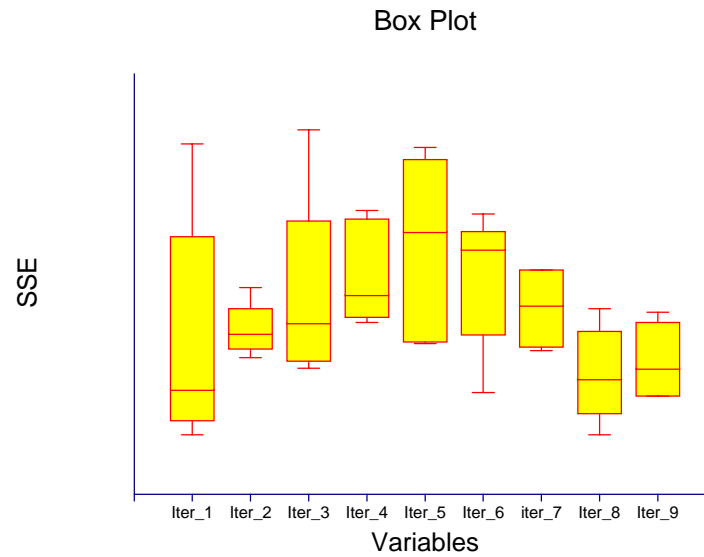


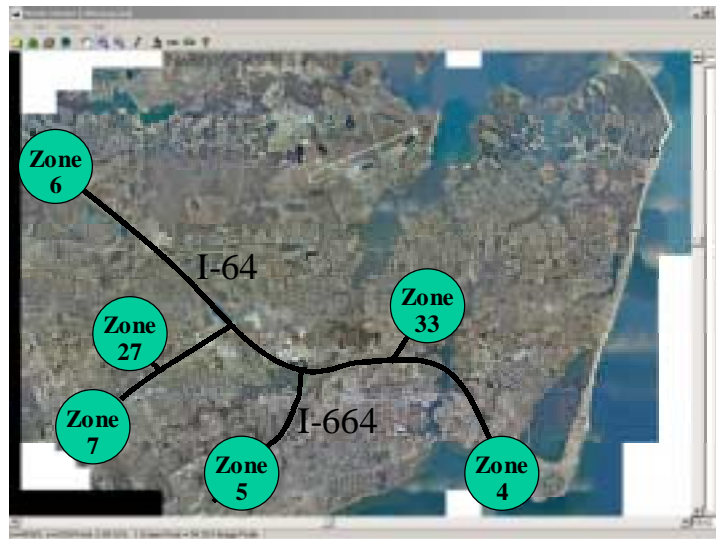
Figure 24. Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS

The SSE measures of estimated and actual link volumes from QUEENSOD iterations and its PARAMICS evaluation show a sizeable discrepancy. There are two reasons for this discrepancy: i) the use of inappropriate assignment matrix and ii) the method of adjusting the O-D demands in the QUEENSOD. Since the assignment matrix was initially extracted from the PARAMICS simulation run under the uniform O-D matrix, the assignment matrix is unable to replicate realistic drivers' route choice behaviors. Secondly, The QUEENSOD method always produces better SSE measures with higher total O-D demand. This is because the QUEENSOD method has a tendency to adjust O-D demands by first matching those observed link flows carrying higher volumes. For example, the heavy traffic demands assigning high traffic volumes on I-64 links come from three external zones (i.e., zones 4, 5 and 6 in Figure 25(a)) as they are the main entry points to the City of Hampton. It was observed that the QUEENSOD

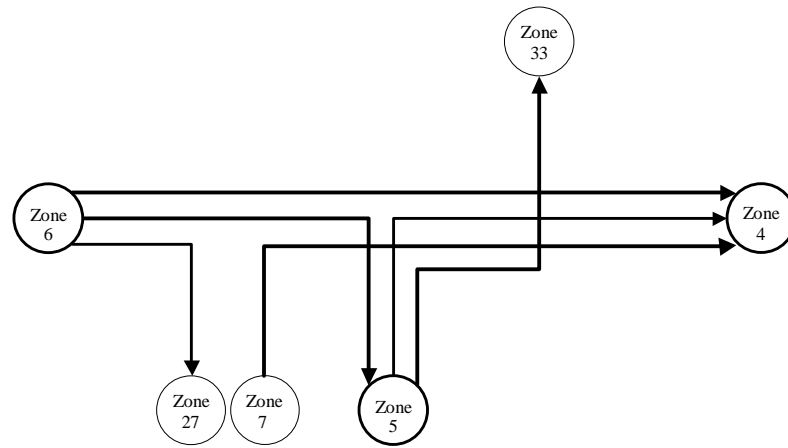
method not only increases O-D demands among these three zones but also increases O-D demands among zones along I-64 highway. This pattern is illustrated in Figure 25. The high demands from zone 6 (I-64, west city limit) and zone 4 (I-64, east city limit) can be easily understood. However, as the number of iterations increases, the demands from zone 7 to zone 4 and zone 5 to zone 33 are increased to match the high flows on I-64 instead of the increasing the demand from zone 6 to zone 4 as shown Figure 25.

Animations show that the estimated O-D demands from iterations 8 and 25 cause a great deal of congestion due to heavy turning volumes that exceed the capacity of the ramps.

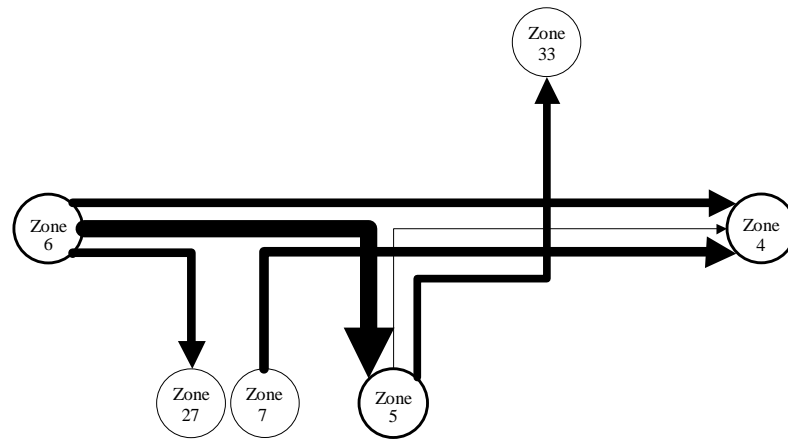
Another observation made during the implementation of the QUEENSOD method is that it increases the O-D demands on zone pairs adjacent to the one that it is trying to match/increase. This pattern of increasing the O-D demands of other zone pairs increases with the number of iterations. This characteristic of the QUEENSOD results in better SSE values for estimates at higher iterations.



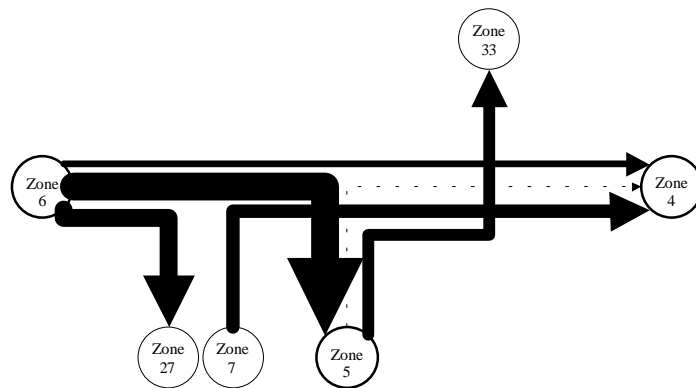
(a) Locations of Zones



(b) O-D Distribution Pattern from Iteration 1



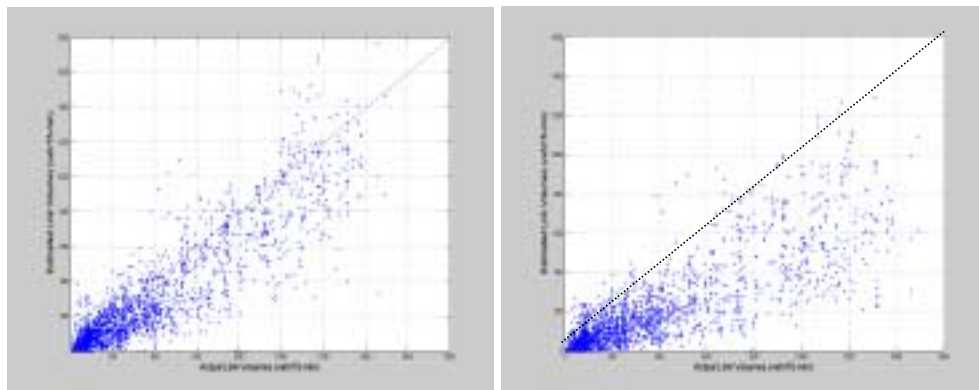
(c) O-D Distribution Pattern from Iteration 8



(d) O-D Distribution Pattern from Iteration 25

Figure 25. O-D Demands Adjustment by the QUEENSOD Model

The O-D demand pattern estimated by the QUEENSOD model causes huge turning demands, particularly on major roads (or high flow links). The abnormally large turning demands result in critical congestions during traffic simulation. In the QUEENSOD iterations, such congestion effects are not considered because the assignment matrix does not consider the capacities of the turning movements. Thus, the estimated link flows from the QUEENSOD iterations and the actual observed link flows can have a high correlation. However, in the evaluation of the O-D estimates from the QUEENSOD method using PARAMICS, the estimated link flows from the PARAMICS simulation and the actual observed link flows showed very small correlation because of congestion effects, in particular, from high turning demands as shown in Figure 26.



(a) QUEENSOD Iteration

(b) Simulation

Figure 26. Comparison of Estimated Link Flows vs. Observed Link Flows

5.3 GA-BASED DYNAMIC O-D ESTIMATION

In the QUEENSOD method, individual 15-minute estimates of dynamic O-D demands at times followed different trends, as shown in Figure 19. The total O-D demands increased through the QUEENSOD iterations, as shown in Figures 20 and 23. Also, the results

between the QUEENSOD iteration and PARAMICS evaluation for them showed big discrepancy, as shown in Figure 26. These results were caused by the local solution that the QUEENSOD-based model found.

In the contrary, GA based on implicit and explicit parallelisms showed its ability to approach global solutions using the schema theorem (19, 25, 27). Also, the promising performances of GA in transportation problems were introduced through researches (6, 7, 21, 32, 38). In this study, the GA-based model is designed to avoid the above problems found in the QUEENSOD-based model using the ability to finding better global solution.

As mentioned in Chapter 4, the binary GA with the solution representation 2 is selected for the GA-based dynamic O-D estimation. The GA model uses the following parameters and the initial population has been randomly generated.

- Normalized geometric selection (selection parameter: 0.08)
- Simple crossover (crossover probability: 0.5)
- Simple mutation (mutation probability: 0.05)
- Maximum generation: 50
- Population size: 200
- Fitness value: sum of squared error (SSE)

5.3.1 Finding Better Assignment Matrix (Step #1)

In the first step, the six different total O-D demands are used in the GA-based O-D estimation. It is noted that the assignment matrix used in the first step GA optimization is same as that of QUEENSOD estimation. The total O-D demand ranges varied from 40,000 to 65,000 vehicles per hour. The results of GA runs are summarized in Table 9. It appears that higher total O-D demand (e.g., 60,000 vehicles per hour) produces better

results. It is noted that during the GA optimization the SSE measures are calculated from assignment matrix and estimated OD matrix. However, the performance of the final O-D matrix estimated from GA optimization is evaluated through the PARAMICS simulation runs. Each O-D matrix was simulated five times with different random number seeds and the average, median and standard deviation of the PARAMICS runs are summarized in Table 10 and illustrated Figure 27. The SSE measures are obtained from observed actual link flows and estimated link flows from the PARAMICS simulation runs. The total O-D demand of 45,000 vehicles per hour provided the best result (median equals to $3.96\text{E}+07$). It is noted that the average SSE value from GA optimization run ($2.86\text{E}+07$, Table 9 column under step 1) and the average SSE value from the PARAMICS simulation runs ($4.09\text{E}+07$, Table 10 column under step 1) exhibit significant differences. This is due to the assignment matrix used in the GA optimization. The initial assignment matrix was obtained from the PARAMICS simulation using a uniform O-D matrix such that it does not represent drivers' route choice behavior.

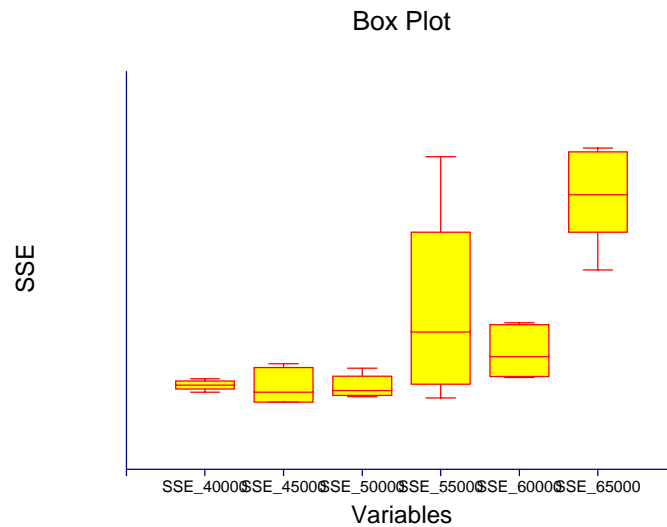


Figure 27. Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS

5.3.2 Finding Appropriate Total O-D Demand (Step #2)

In the second step, a new assignment matrix is extracted from the PARAMICS simulation run under the best O-D matrix which was found at the total O-D demand of 45,000 vehicles per hour. The GA optimization runs were again conducted from total O-D demand between 40,000 to 60,000 vehicles per hour. Note that the total O-D demand of 65,000 vehicles per hour was eliminated as it produced extremely high SSE measure ($8.96\text{E}+07$, Table 10). The results of GA optimization runs were shown in Table 9 under step 2 column. The PARAMICS simulation results under these O-D demands are shown in Figure 28 and Table 10 under step 2 column. Again, the O-D matrix estimated under the total O-D demand at 45,000 vehicles per hour produced the lowest SSE measure of $3.78\text{E}+07$ as shown in Table 10. Even though the SSE discrepancy between GA optimization and PARAMIC simulations is reduced from that of step 1, it is still very big. Thus, the step 3 runs are conducted.

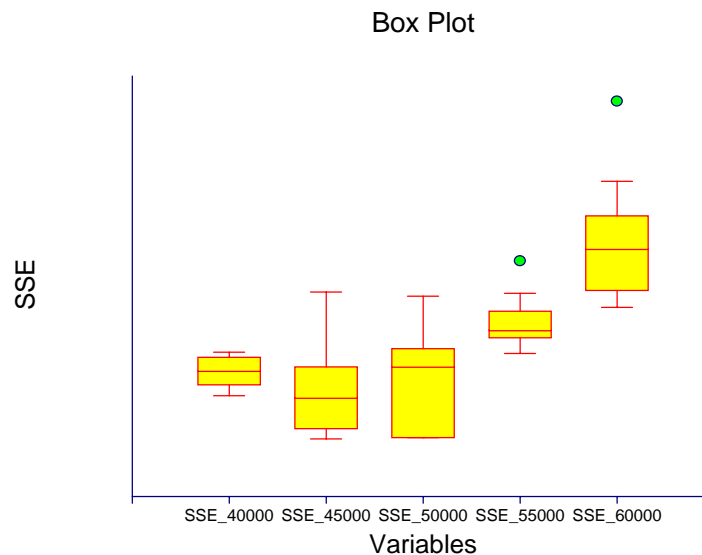


Figure 28. Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS

5.3.3 Finding Best Total O-D Demand and O-D Matrix (Step #3)

During the step 3, three total O-D demands of 42,500, 45,000, and 47,500 vehicles per hour are considered as the best O-D demand was found at 45,000 in step 2. Figure 29 illustrates the convergence of the GA model with total O-D demand of 45,000 vehicle/hour.

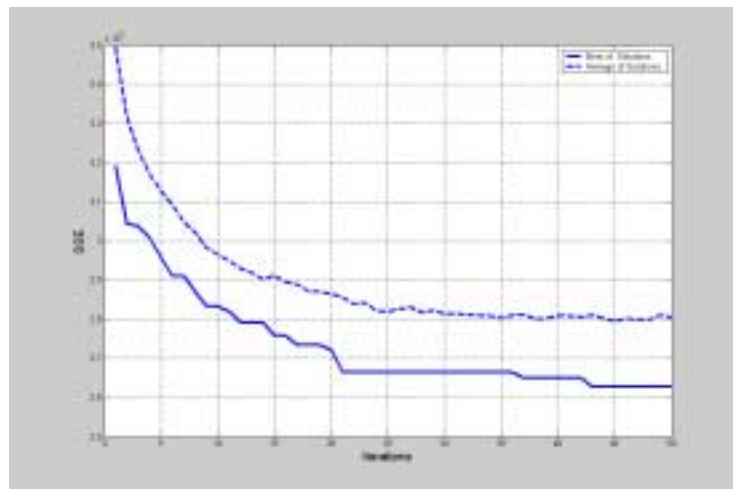


Figure 29. Convergence of GA Iterations

The new assignment matrix was extracted from the best PARAMICS simulation run and used in the GA optimization runs. The SSE measures from the GA optimization and the PARAMICS simulation runs are shown in Tables 9 and 10, respectively.

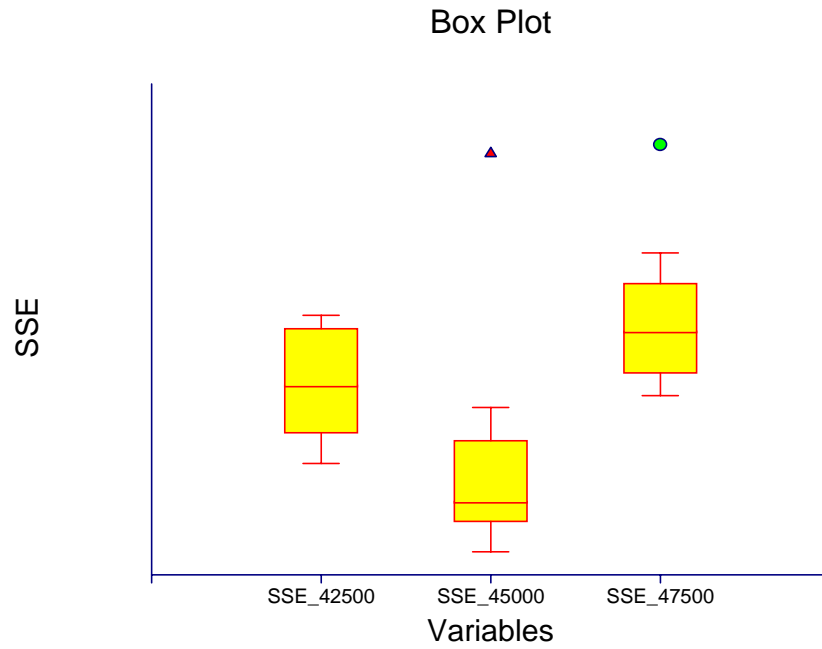


Figure 30. Box Plot of O-D Matrix Estimates Evaluation Using PARAMICS

Table 9. Results of the GA Runs

Total Sum of O-D Demands (veh./hour)	SSE		
	Step 1	Step 2	Step 3
40,000	3.29E+07	3.17E+07	-
42,500	-	-	2.68E+07
45,000	2.86E+07	2.77E+07	2.63E+07
47,500	-	-	2.45E+07
50,000	2.56E+07	2.47E+07	-
55,000	2.28E+07	2.22E+07	-
60,000	2.13E+07	2.04E+07	-
65,000	2.30E+07	-	-

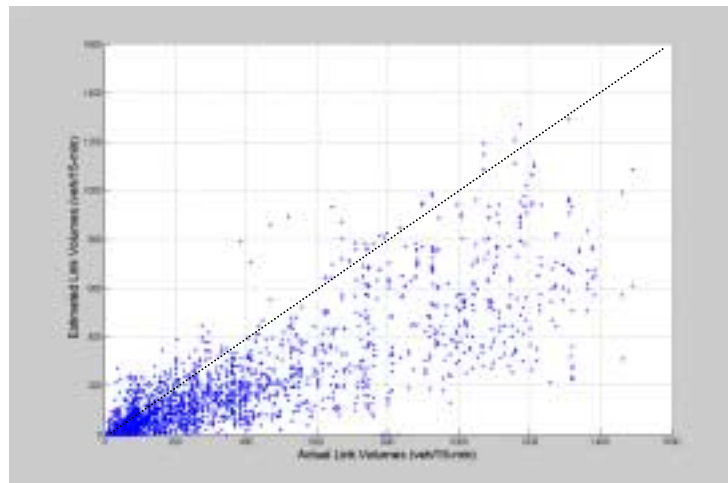
Table 10. Evaluation Results of Estimated O-D Matrix from PARAMICS

Total Sum of O-D Demands (veh./hour)	Statistics	SSE		
		Step 1	Step 2	Step 3
40,000	Average	4.12E+07	3.90E+07	-
	Median	4.15E+07	3.90E+07	-
	STD	1.24E+06	1.06E+06	-
42,500	Average	-	-	2.99E+07
	Median	-	-	2.99E+07
	STD	-	-	1.07E+06
45,000	Average	4.09E+07	3.78E+07	2.83E+07
	Median	3.96E+07	3.71E+07	2.75E+07
	STD	4.44E+06	3.52E+06	2.36E+06
47,500	Average	-	-	3.13E+07
	Median	-	-	3.10E+07
	STD	-	-	1.51E+06
50,000	Average	4.08E+07	3.84E+07	-
	Median	4.01E+07	3.93E+07	-
	STD	2.82E+06	3.47E+06	-
55,000	Average	5.94E+07	4.25E+07	-
	Median	5.48E+07	4.19E+07	-
	STD	2.36E+06	1.92E+06	-
60,000	Average	4.96E+07	4.82E+07	-
	Median	4.86E+07	4.77E+07	-
	STD	6.56E+06	4.44E+06	-
65,000	Average	8.96E+07	-	-
	Median	8.92E+07	-	-
	STD	1.21E+07	-	-

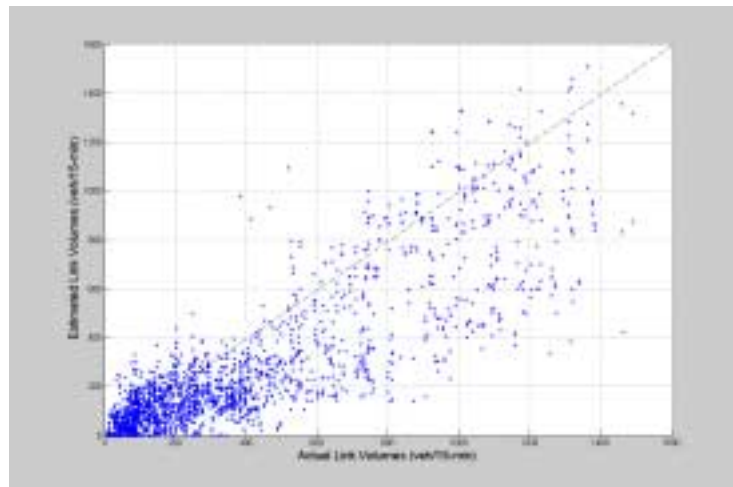
The PARAMIC simulation results indicate that the O-D matrix under 45,000 vehicles per hour O-D demand produced the best estimates. The SSE discrepancy between the GA optimization ($2.63\text{E}+07$) and the PARAMICS simulation runs ($2.83\text{E}+07$) were significantly reduced. Here, we can watch that the third step further reduced the discrepancy.

5.4 SUMMARY

As shown in Tables 8 and 10, the SSE measures based estimated link flows from the PARAMICS simulation and actual observed link counts indicate that GA method outperforms QUEENSOD method. The results of estimated OD and observed actual comparisons of GA-based method and QUEENSOD method are shown in Figure 31.



(a) QUEENSOD method (SSE = $4.61\text{E}+07$)



(b) GA method ($SSE = 2.83E+07$)

Figure 31. Comparison of Estimated Link Flows vs. Observed Link Flows

Note: the estimated link flows obtained from the PARAMICS simulation runs under optimal O-D matrix

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 CONCLUSIONS

In this study, a development of ITS test-bed using microscopic traffic simulation was presented. The major contributions of this study are i) presenting traffic simulation network building process and the efforts to improve its reliability, ii) development of API for coordinated actuated signal control, iii) coupling dynamic O-D estimation and traffic simulation model, and iv) proposing a new approach based on GA for the dynamic O-D estimation problem.

The proposed methodology was demonstrated through a case study for a City of Hampton network. The test-bed development involved three major steps. These are building a basic traffic simulation network, the API for coordinated actuated signal control, and the dynamic O-D estimation for the network.

During the network building, lots of unexpected problems such as unrealistic behaviors of individual vehicle in specific locations and unacceptable huge congestions in ramps or intersections were encountered. These were identified through simulation model visualization. Based on the visualization, these problems were removed by adjusting changing “next lane” setting and/or parameters for “signposting.” It was concluded that the simulation model need to be verified using visualization before implementing for any evaluations.

As the PARAMICS program does not provide coordinated actuated signal control, an API was developed as part of the simulation test-bed for ITS evaluation. The process of API development involved the use of dummy phases to realize NEMA dual-ring control. It was also found that the yellow time interval in the PARAMICS program is a network variable such that each intersection could not have varying yellow time interval. Furthermore, whenever a phase skip occurs, yellow time of that phase was not recalled. Thus, the proposed API uses effective green time such that issue of yellow time was resolved.

One of the critical elements in the implementation of a large-scale simulation model is to use an appropriate O-D matrix. Two practical methods, QUEENSOD and genetic algorithm, were considered. The GA-based O-D estimation model was developed in this project. A series of evaluation runs were made to obtain optimal GA parameters, representations and settings. The QUEENSOD method was implemented as is practiced in the literature. One enhancement made in this research during O-D matrix estimation is the use of assignment matrix. The O-D estimation was conducted over a series of steps. Each step develops optimal O-D matrix for a given assignment matrix. The assignment

matrix was updated through a microscopic simulation program using an optimal O-D found at the end of each step. Thus, both O-D matrix and assignment matrix are updated until the discrepancies between estimated link volumes from assignment matrix and microscopic simulation are acceptable. Note that both link volumes were produced from an optimal O-D at the end of each step. The results of two O-D estimation models indicated that the GA-based model outperformed QUEENSOD model. It was shown that the GA-based dynamic O-D estimation method has greater potential for estimating a better O-D matrix for a large scale and congested network. In the case study, the QUEENSOD method showed some limitations in finding a global solution.

6.2 RECOMMENDATIONS

Based on the findings and lessons learned during this research, the following recommendations were made:

1. It appears that microscopic simulation programs play an important role in the evaluation and testing of various ITS functions and other applications. It is recommended that the test-bed developed in this research be used in comparison of alternatives, estimation of impacts and sensitivity analysis for ITS deployments related to the following ITS user services: pre-trip travel Information, route guidance, traffic control, travel demand management, emissions testing and mitigation, and so on. In addition, it can evaluate evacuation scenarios and signal coordination for the entire city.

2. The dynamic O-D estimation methods used in this study utilize a multi-step approach. Since the O-D estimation is assumed to start from a uniform O-D matrix, the first step is intended to find a reasonable assignment matrix. Through the case study using both the GA method and the QUEENSOD method, the use of improved assignment matrix, in which estimated after first step, showed better dynamic O-D estimations.
3. The case study conducted in this research should be useful in the development of similar a simulation-based test-bed because the lessons learned could reduce trial-and-errors and efforts needed. Especially, the newly developed API for coordinated and actuated signal control could be applied to other signalized intersections with a minimal updates.
4. Both the GA and the QUEENSOD methods only used link volume information. Future study should consider the use of turning movement counts such that the discrepancies between estimated and observed link counts be quickly reduced. This could be achieved by adjusting the fitness function in the GA model. Furthermore, the GA model could include network delays or congestion levels for its criteria for dynamic O-D estimation by changing its evaluation function.

REFERENCES

1. White, Timothy. General overview of Simulation Models. *49th Annual Meeting, Southern District Institute of Transportation Engineers*, Williamsburg, Virginia, April 22-25, 2001.
2. Park, B., N. M. Roupail, and J. Sacks. Assessment of a Stochastic Signal Optimization Method using Microsimulation. *Accepted for publication in the Transportation Research Record*, 2001.
3. Park, B. and I. Yun. Evaluation of Microscopic Simulation Programs for Coordinated Signal System. *13th ITS America's Annual Meeting*, May 19 – 22, 2003
4. Ortuzar, J. D. and L. G. Willumsen. *Modeling Transport*. Wiley & Sons, 1994.
5. Euse, S. M. Urban Transportation in Practice, 1: Conventional Analysis. *J. Transp. Engrg.*, ASCE, 119(6), pp. 793-815, 1993.
6. Kim, H., S. Baek and Y. Lim. O-D Matrices Estimation Using Genetic Algorithm From Traffic Counts. *Transportation Research Record 1771*, TRB, National Research Council, Washington, D.C. 2000.
7. Yin, Y. Genetics Algorithm Based Approach for Bilevel Programming Models, *Journal of Transportation Engineering*, March/April 2000, pp. 115 – 120.
8. Paramahamsan, H. *Fundamental Properties of Synthetic O-D Generation Formulations and Solutions*. Thesis for M.S. Degree, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, January 1999.
9. Ashok, Kalidas. Estimation and Prediction of Time-Dependent Origin-Destination Flows. Ph.D. Thesis, MIT, September 1996.
10. Cascetta, Ennio and Domenico Inaudi, and Gerald Marquis. Dynamic Estimators of Origin-Destination Matrices using Traffic Counts. *Transportation Science*, 27(4), pp. 363-373, 1993.
11. Hellinga, B. R. and M. Van Aerde, Estimating Dynamic OD Demands for a Freeway Corridor Using Loop Detector Data. *Proceedings of the Canadian Society for Civil Engineering 1998 Annual Conference held in Halifax, Nova Scotia*. Volume IVb, p. 185 - 197. ISBN 0-921303-86-6.

12. Van Aerde, M., B. Hellinga and G. MacKinnon. QUEENSOD: A Method for Estimating Time Varying Origin-Destination Demands For Freeway Corridores/Networks. *Presented at Annual TRB Meeting*, Washington D.C., January 1993.
13. Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems, *Journal of Basic Eng.*, Trans. ASME, Series D 82(1):33-45, 1960.
14. Okutani, I. The Kalman Filtering Approaches in Some Transportation and Traffic Problems. In N. H. Gartner and N. H. M. Wilson (eds), *Proceedings of the Tenth International Symposium on Transportation and Traffic Theory*, Elsevier, New-York, pp. 397-416, July 8-10, 1987.
15. Ashok, K. and M. Ben-Akiva, Dynamic Origin-Destination Matrix Estimation and Prediction for Real-Time Traffic management Systems, In C.F. Daganzo, editor, *International Symposium on Transportation and Traffic Theory*, 465-484, Elsevier Science Publishing Company, Inc. 1993.
16. Bierlaire, M. and F. Crittin. An Efficient Algorithm For Real-time Estimation and Prediction of Dynamic OD Tables, *Submitted to Operation Research*, Report # RO-010808 ROSO-DMA-EPFL, August 8. 2001.
17. He, Rachel R., Alain L. Kornhauser, Bin Ran, Estimation of Time-dependent O-D Demands and Route Choice form Link Flows, *Presented at TRB 2002 Annual Meeting*, Washington D.C., February 2002.
18. Taekeris, Theodore and Antony Stathopoulos, Real-time Dynamic Matrix Adjustment Using Simulated and Actual Link Flows in Urban Networks, *Presented in 2003 TRB Annual Meeting*, Washington D.C., February 2003.
19. Hollend, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
20. Whitley, Darrell, *A Genetic Algorithm Tutorial*, Colorado State University.
21. Park, Byungkyu, *Development of Genetic Algorithm-based Signal Optimization Program for Oversaturated Intersections*, Ph.D. Thesis, Department of Civil Engineering, Texas A&M University, August 1998.
22. Davis, L. *The Handbook of Genetic Algorithm*. Van Nostrand Reingold, New York, 1991.

23. Goldberg, J. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
24. Houck, R. Christopher, Jeffery A. Joines and Michael G. Kay, *A Genetic Algorithm for Function Optimization: A MATLAB Implementation*. North Carolina State University.
25. Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, 1994.
26. Dasgupta, D. and Z. Michalewicz. Evolutionary Algorithms – An Overview. In *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, New York, 1997.
27. Beasley, D., D. R. Bull, R. R. Martin. An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, Vol. 15. No.2, 58-69, 1994.
28. Yang, H., Y. Iida and T. Sasaki. Estimation of O-D Matrix from Traffic Counts on Congested Networks, *Transportation Research 26B*, 1992. pp. 417-434.
29. *Modeller V3.0 User Guide*, QUADSTONE Ltd., Edinburgh, UK, 2001.
30. *Programmer V3.0 User Guide*, QUADSTONE Ltd., Edinburgh, UK, 2001.
31. Liu, Henry X., Lianyu Chu, and Will Recker. *Paramics API Development Document for Actuated Signal, Signal Coordination and Ramp Control*. California PATH Working Paper, UCB-ITS-PEP-2001-11, University of California, Irvine, February 2001.
32. Tao Ma, MASc and Baher Abdulhai, GENOSIM: A Genetic Algorithm-Based Optimization Approach and Genetic Tool for the Calibration of Traffic Microscopic Simulation Parameters, *Presented at TRB 2002 Annual Meeting*, Washington, D.C., January 2002.
33. Garber, Nicholas J. and Lester A. Hoel. *Traffic and Highway Engineering, Revised Second Edition*. PWS Publishing, 1996.
34. *SYNCHRO Version 5 User Guide*. Trafficware Corporation, Albany, California, 2001.
35. Hu, Shou-ren. An Adaptive Kalman Filtering Algorithm for the Dynamic Estimation and Prediction of Freeway Origin-Destination Matrices. Ph.D. Thesis, Purdue University, December 1996.

36. Van Aerde, Michel, Hesham Rakha and Harinarayan Paramahamsan. Estimation of O-D Matrices: The Relationship between Practical and Theoretical Considerations. *Presented at 82nd TRB Annual Meeting*, Washington D.C., January 2003.
37. Rakha, H., M. Van Aerde, L. Bloomberg, and X. Huang. Construction and Calibration of a Large-scale Microsimulation Model of the Slat Lake Area. *Transportation Research Record 1644*, TRB, National Research Council, Washington D.C., 1998.
38. W. Sadek, B. L. Smith and M. J. Demetsky. Dynamic Traffic Assignment: A Genetic Algorithms Approach. *Transportation Research Record 1588*, TRB, National Research Council, Washington, D.C. 1997.
39. Nelson, Eric J. and Darcy Bullock. Impact of Emergency Vehicle Preemption on Signalized Corridor Operation: An Evaluation. *Transportation Research Record 1727*, TRB, National Research Council, Washington, D.C. 2000.
40. Hansen, Blake G., Peter T. Martin, and H. Joseph Perrin, Jr. SCOOT Real-Time Adaptive Control in a CORSIM Simulation Environment. *Transportation Research Record 1727*, TRB, National Research Council, Washington, D.C. 2000.
41. Lucas, David E., Pitu B. Mirchandani, and K. Larry Head. Remote Simulation to Evaluate Real-Time Traffic Control Strategies. *Transportation Research Record 1727*, TRB, National Research Council, Washington, D.C. 2000.
42. Jayakrishnan, R., Jun-Seok Oh, and Abd-El-Kader Sahraoui. Calibration and Path Dynamic Issues in Microscopic Simulation for Advanced Traffic management and Information Systems. *Transportation Research Record 1771*, TRB, National Research Council, Washington, D.C. 2000.
43. Lianyu Chu, Henry X. Liu, Will Recker, and Steve Hague. *Evaluation of the Effectiveness of Potential ATMIS Strategies Using Microscopic Simulation*. Center for Traffic Simulation Studies. Paper UCI-ITS-TS-WP-02-13, August 1, 2002.

APPENDIX A - PHASE SEQUENCE OF 25 ORDERS IN API

Table A. Phase Sequences by Order Number in API

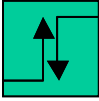
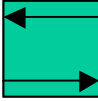
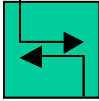
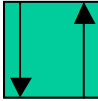
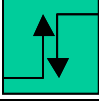
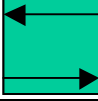
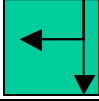
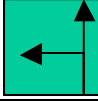
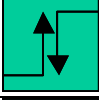
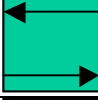
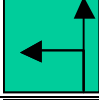
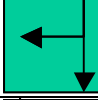
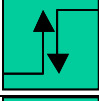
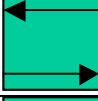
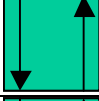
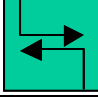
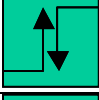
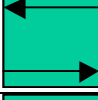
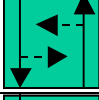
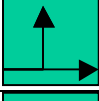

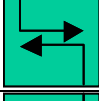
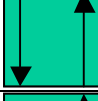
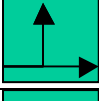

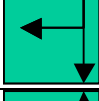
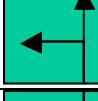
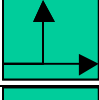

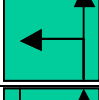
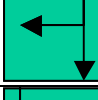
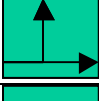
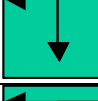
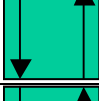
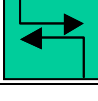
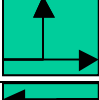
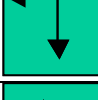
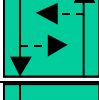
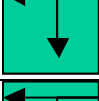
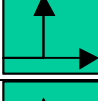
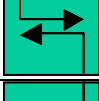
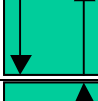
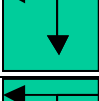
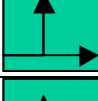
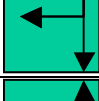
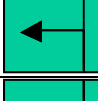
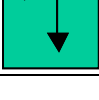
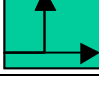

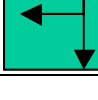
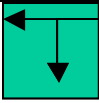

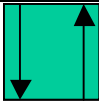
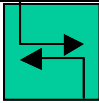
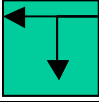
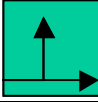

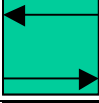
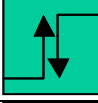
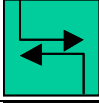

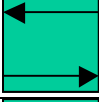
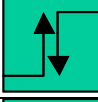
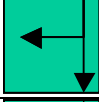
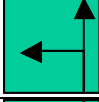
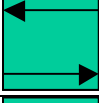
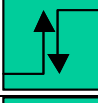
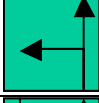
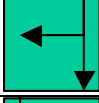
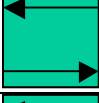

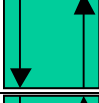
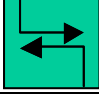
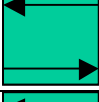
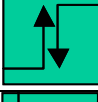
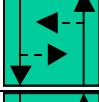

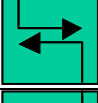
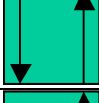

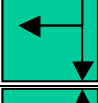
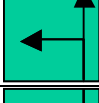

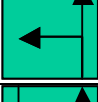
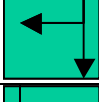
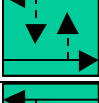
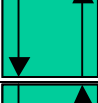
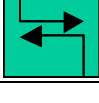


Order Num.	Phase #1	Phase #2	Phase #3	Phase #4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

Table A. Phase Sequences by Order Number in API (continuous)

Order Num.	Phase #1	Phase #2	Phase #3	Phase #4
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				

* Overlap phases are not represented in this table.

APPENDIX B - INPUT FILES OF API FOR COORDINATED ACTUATED SIGNAL CONTROLLER

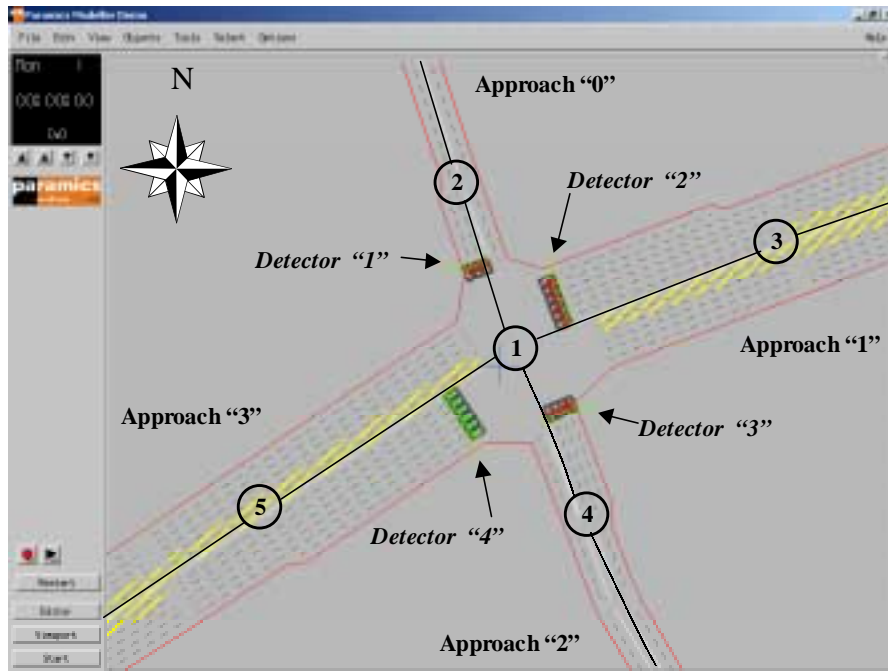


Figure B-1. Layout of Example Intersection

Appendix B-1. Example of “priorities” File for Order #1

```

actions 1
phase offset 0.00 sec
phase 1
    75.00
    max 180.00
red phase 3.00
fill
all barred except
from 3 to 5 major
from 5 to 3 major
from 3 to 2 minor
from 5 to 4 minor
phase 2
    11.00
    max 11.00
red phase 3.00
fill
all barred except
from 2 to 3 major
from 4 to 5 major
from 3 to 2 minor
from 5 to 4 minor
phase 3
    0.00
    max 11.00
red phase 0.00
fill
all barred except
from 2 to 3 major
from 2 to 4 major
from 2 to 5 minor
from 3 to 2 minor
phase 4
    0.00
    max 3.00
red phase 0.00
fill
all barred except
from 2 to 4 major
from 2 to 5 minor
from 3 to 2 minor
phase 5

```

5.00
 max 19.00
 red phase 3.00
 fill
 all barred except
 from 4 to 2 major
 from 4 to 5 major
 from 4 to 3 minor
 from 5 to 4 minor
 phase 6
 0.00
 max 3.00
 red phase 0.00
 fill
 all barred except
 from 4 to 2 major
 from 4 to 3 minor
 from 5 to 4 minor
 phase 7
 47.00
 max 180.00
 red phase 3.00
 fill
 all barred except
 from 2 to 4 major
 from 4 to 2 major
 from 2 to 5 minor
 from 4 to 3 minor
 phase 8
 15.00
 max 180.00
 red phase 3.00
 fill
 all barred except
 from 3 to 4 major
 from 5 to 2 major
 from 2 to 5 minor
 from 4 to 3 minor
 phase 9
 0.00
 max 180.00
 red phase 0.00
 fill
 all barred except
 from 5 to 2 major

from 5 to 3 major
 from 2 to 5 minor
 from 5 to 4 minor
 phase 10
 0.00
 max 3.00
 red phase 0.00
 fill
 all barred except
 from 5 to 3 major
 from 2 to 5 minor
 from 5 to 4 minor
 phase 11
 9.00
 max 180.00
 red phase 3.00
 fill
 all barred except
 from 3 to 4 major
 from 3 to 5 major
 from 3 to 2 minor
 from 4 to 3 minor
 phase 12
 0.00
 max 3.00
 red phase 0.00
 fill
 all barred except
 from 3 to 5 major
 from 3 to 2 minor
 from 4 to 3 minor
 phase 13
 0.00
 max 77.00
 red phase 0.00
 fill
 all barred except
 from 3 to 5 major
 from 5 to 3 major
 from 3 to 2 minor
 from 5 to 4 minor

Appendix B-2. Input Data for Order #1

```

struct intersections
{
    int node_name;        //Node number used in PARAMICS Zoom Window
    int node_n;           //Node number used in API
    void *Nodep;          //Point variable for node
    int order;            //Phase order number
    int cycle;            //Cycle length
    int major_dir;        //East and westbound: 1, north and southbound: 0
    int major[2];         //Major approach numbers
    int minor[2];         //Minor approach numbers
    int n_approach;       //The number of approaches
    int detector_ID[3];   //Detector IDs
    int n_left_lane;      //The number of left lanes
    int left_lanes[3];    //Left lane number
    int n_through_lane;   //The number of through lanes
    int through_lanes[5]; //Through lane number
    int NEMA[13][2];      //NEMA Phase numbering of movements used in 13 PARAMICS Phases
    int NEMA_app[2][8];   //Approach number for each NEMA phase number according to "major_dir"
    int n_actual_p;       //The number of phases actual and overlap phases
    int n_used_p;         //The number of all phases
    int actual_p[13];     //Actual phase number
    int next_p[13];       //Next phase number
    int n_p_app[13];      //The number of approaches that use each phase
    int p_app[13][2];     //Approach number that use each phase
    int overlap[13];      //Indicator for overlap phase
    float stored_g[13];   //stored green time for each phase
    float min_g[13];     //minimum green time for each phase
    float max_g[13];     //maximum green time for each phase
    float stored_r[13];   //stored all-red time for each phase
    float proper_r[13];   //All-red time to be assigned to dummy phases
    float ext[13];        //Extension time
    int cur_p;            //Current phase number
    int pre_p;            //Previous phase number
    int ori_p;            //Original phase number during extension of the coordinated phase
    int cur_time;         //Current simulation time
    float gaptime_l[4];   //gap time for four left-turns
    int call_l[4];        //Existence of demand call for four left-turns
    float gaptime_t[4];   //gap time for four through movements
    int call_t[4];        //Existence of demand call for four through movements
    float forceoff[13];   //Thirteen Force-off points
    float no_permissive_time[13]; //Length of permissive period times for 13 phases
    int first_run;        //flag
    int is_first_r;       //flag
    int actuated[3];      //flag
    int permissive;       //flag
    int is_permissive;    //flag
    int per_first_time;   //flag
};

```

[illegible]

APPENDIX C - SOLUTION DECODING CONCEPTS IN THE GA

Appendix C-1. Decoding Example for Solution Representation 1

$$T[1][2][3] = T \times \frac{x[3]}{\sum_{dt} x[dt]} \times \frac{y[1][3]}{\sum_i y[i][3]} \times \frac{z[1][2][3]}{\sum_j z[1][j][3]} \quad (1)$$

where,

- | | | |
|---------------|---|--|
| T | = | given total O-D demand, |
| $T[i][j][dt]$ | = | trip leaving origin i to destination j during departure time interval dt , |
| $x[dt]$ | = | parameters in the first group of a solution, the parameters are used to divide a given total O-D demand into four total O-D demands during each four 15-minute departure time intervals (dt), the number of the parameters is equal to n_{dt} . |
| $y[i][dt]$ | = | parameters in the second group of a solution, the parameters are used to divide four total 15-minute interval O-D demands into the total of trips leaving from i during dt , the number of the parameter is equal to $n_i \times n_{dt}$, |
| $z[i][j][dt]$ | = | parameters in the third group of a solution, $i \neq j$, the parameters are used to divide the total of trips generated from i during dt into trips leaving from i to j during dt , the number of the parameters is equal to $n_i \times (n_j - 1) \times n_{dt}$, |
| i | = | origin, $1 < i < n_i$ (the number of zones: 50), |
| j | = | destination, $1 < j < n_j$ (the number of zones: 50), and |
| dt | = | departure time interval, $1 < dt < n_{dt}$ (the number of departure time intervals: 4). |

Appendix C-2. Decoding Example for Solution Representation 2

- Trips generated from external zones

$$T[1][2][3] = \tilde{T}[1] \times \frac{y[1][2]}{\sum_j y[1][j]} \times z[1][2][3] \quad (2)$$

- Trips generated from internal zones

$$T[8][2][3] = \left(T - \sum_{i \in EZ} \tilde{T}[i] \right) \times \frac{x[8]}{\sum_{i \in IZ} x[i]} \times \frac{y[8][2]}{\sum_j y[8][j]} \times z[8][2][3] \quad (3)$$

where,

$\tilde{T}[i]$	=	observed trip leaving origin i , $i \in EZ$,
$x[i]$	=	parameters in the first group of a solution, the parameters are used to divide a given total O-D demand into the total of trips leaving from i , the number of the parameter is equal to $n_i - n_{EZ}$,
$y[i][j]$	=	parameters in the second group of a solution, the parameters are used to divide trips leaving from i into trips leaving from i to j , the number of the parameter is equal to $n_i \times (n_j - 1)$,
$z[i][j][dt]$	=	parameters in the third group of a solution, $i \neq j$, the parameters are used to divide trips leaving from i to j into trips leaving from i to j during each four 15-minute departure time intervals (dt), the number of the parameters is equal to $n_i \times (n_j - 1) \times (ndt - 1)$, $dt=1, 2$, and 3 , where, $z[i][j][4] = 1.0 - (z[i][j][1] - z[i][j][2] - z[i][j][3])$,
EZ	=	external zones, $EZ=1, \dots, n_{EZ}$, n_{EZ} is the number of external zones (7), and
IZ	=	internal zones, $IZ=8, \dots, 50$,

Appendix C-3. Decoding Example for Solution Representation 3

- Trips generated from external zones

$$T[1][2][3] = \tilde{T}[1][3] \times \frac{z[1][2][3]}{\sum_j z[1][j][3]} \quad (4)$$

- Trips generated from internal zones

$$T[8][2][3] = \left(T \times \frac{x[3]}{\sum_{dt} x[dt]} - \sum_{i \in EZ} \tilde{T}[i][3] \right) \times \frac{y[8][3]}{\sum_i y[i][3]} \times \frac{z[8][2][3]}{\sum_j z[8][j][3]} \quad (5)$$

where,

- | | | |
|--------------------|---|---|
| $\tilde{T}[i][dt]$ | = | observed trip leaving origin i during dt , $i \in EZ$, |
| $x[dt]$ | = | parameters in the first group of a solution, the parameters are used to divide a given total O-D demand into four total O-D demands during each four 15-minute departure time intervals (dt), the number of the parameters is equal to ndt , |
| $y[i][dt]$ | = | parameters in the second group of a solution, the parameters are used to divide four total 15-minute interval O-D demands into the total of trips leaving from i during dt , the number of the parameter is equal to $(n_i - n_{EZ}) \times ndt$, and |
| $z[i][j][dt]$ | = | parameters in the third group of a solution, $i \neq j$, the parameters are used to divide the total of trips generated from i during dt into trips leaving from i to j during dt , the number of the parameters is equal to $n_i \times (n_j - 1) \times ndt$. |

APPENDIX D - MATLAB CODES FOR QUEENSOD-BASED MODEL AND GA-BASED MODEL

Appendix D-1. MATLAB Code for QUEENSOD-based Model

```
%Name: D10_step2.m
%DynamicIter_Hampton_MDLUP.m
%Maker: ILSOO YUN
%Contents: QueensOD Method for time-varying OD demand for Hampton

clc
clear

Files=['Results01';'Results02';'Results03';'Results04';'Results05';'Results06';'Results07';'Results08';'Results09';'Results10';'Results11';'Results12';'Results13';'Results14';'Results15';'Results16';'Results17';'Results18';'Results19';'Results20';'Results21';'Results22';'Results23';'Results24';'Results25';'Results26';'Results27';'Results28';'Results29';'Results30';'Results31';'Results32';'Results33';'Results34';'Results35';'Results36';'Results37';'Results38';'Results39';'Results40';'Results41';'Results42';'Results43';'Results44';'Results45';'Results46';'Results47';'Results48';'Results49';'Results50'];

load('OD.mat');
load('MDLUP.mat');
load('VA.mat');
load P_MDLUP
load I_MDLUP

CO=[254 345 296 260;
    389 374 421 374;
    258 269 250 250;
    835 757 798 789;
    1061 1005 943 890;
    1001 1099 1070 993;
    1191 1085 1047 944];

tic
%variable explanation
%o: Origin
%d: Destination
%dt: Departure time; total is 4 in this case
%t: Time Period; total is 5 in this case

%Input variables explanation
%DLUP(o,d,dt,k,t1-5);
%OD(o,d,dt)
%VA(k,t1-t5)

%Initialization (First, I will test the test network and data set)
n_o=50;n_d=50;n_dt=4;n_k=3340;n_t=8;PTR=5000;

[low_DLUP col_DLUP]=size(MDLUP);

CL=zeros(n_k,n_t);
```

```

CS=zeros(n_o,n_d,n_dt);
PO=zeros(n_o,n_d,n_dt);
CF=zeros(n_o,n_d,n_dt);
NewOD=zeros(n_o,n_d,n_dt);
EV1=zeros(100,4);
EV2=zeros(100,4);
disp('Calculating PO')
for(o=1:n_o)
    for(d=1:n_d)
        if(o~=d)
            for(dt=1:n_dt)
                tt=P_MDLUP(1:P_MDLUP(PTR,I_MDLUP(o,d,dt)),I_MDLUP(o,d,dt));
                % tt=find(MDLUP(:,1)==o & MDLUP(:,2)==d & MDLUP(:,3)==dt);
                PO(o,d,dt)=sum(sum(MDLUP(tt,5:12)));
            end
        end
    end
end

for(iter=1:50)
    iter
    VE=zeros(n_k,n_t);

    disp('Calculating temp VE')
    for(o=1:n_o)
        for(d=1:n_d)
            if(o~=d)
                for(dt=1:n_dt)%
                    if(iter==1)
                        tt=MDLUP(P_MDLUP(1:P_MDLUP(PTR,I_MDLUP(o,d,dt)),I_MDLUP(o,d,dt)),:);
                        if(~isempty(tt))
                            [a b]=size(tt);
                            for(iter_tt=1:a)
                                tt(iter_tt,5:12)=tt(iter_tt,5:12).*OD(o,d,dt);

                                VE(tt(iter_tt,4),1)=tt(iter_tt,5)+VE(tt(iter_tt,4),1);
                                VE(tt(iter_tt,4),2)=tt(iter_tt,6)+VE(tt(iter_tt,4),2);
                                VE(tt(iter_tt,4),3)=tt(iter_tt,7)+VE(tt(iter_tt,4),3);
                                VE(tt(iter_tt,4),4)=tt(iter_tt,8)+VE(tt(iter_tt,4),4);
                                VE(tt(iter_tt,4),5)=tt(iter_tt,9)+VE(tt(iter_tt,4),5);
                                VE(tt(iter_tt,4),6)=tt(iter_tt,10)+VE(tt(iter_tt,4),6);
                                VE(tt(iter_tt,4),7)=tt(iter_tt,11)+VE(tt(iter_tt,4),7);
                                VE(tt(iter_tt,4),8)=tt(iter_tt,12)+VE(tt(iter_tt,4),8);
                            end
                        end
                    else
                        tt=MDLUP(P_MDLUP(1:P_MDLUP(PTR,I_MDLUP(o,d,dt)),I_MDLUP(o,d,dt)),:);
                        if(~isempty(tt))
                            [a b]=size(tt);
                            for(iter_tt=1:a)
                                tt(iter_tt,5:12)=tt(iter_tt,5:12).*NewOD(o,d,dt);
                                VE(tt(iter_tt,4),1)=tt(iter_tt,5)+VE(tt(iter_tt,4),1);
                                VE(tt(iter_tt,4),2)=tt(iter_tt,6)+VE(tt(iter_tt,4),2);
                                VE(tt(iter_tt,4),3)=tt(iter_tt,7)+VE(tt(iter_tt,4),3);

```

```

        VE(tt(iter_tt,4),4)=tt(iter_tt,8)+VE(tt(iter_tt,4),4);
        VE(tt(iter_tt,4),5)=tt(iter_tt,9)+VE(tt(iter_tt,4),5);
        VE(tt(iter_tt,4),6)=tt(iter_tt,10)+VE(tt(iter_tt,4),6);
        VE(tt(iter_tt,4),7)=tt(iter_tt,11)+VE(tt(iter_tt,4),7);
        VE(tt(iter_tt,4),8)=tt(iter_tt,12)+VE(tt(iter_tt,4),8);
    end
end
end
end
end
end
end

disp('Calculating VE and CL')
for(k=1:n_k)
    for(t=1:4)
        VE(k,t)=0.973*VE(k,t+4)+VE(k,t);
    end
    for(t=5:6)
        VE(k,t)=VE(k,t)+0.827*VE(k,t-4);
    end
end

for(k=1:n_k)
    for(t=1:6)
        if(VE(k,t)~=0)
            CL(k,t)=VA(k,t)/VE(k,t);
        else
            CL(k,t)=1.0;
        end
    end
end

disp('Calculating CS')
CS_tmp=MDLUP(:,1:10);

for(iter1=1:low_DLUP)
    for(t=1:6)
        CS_tmp(iter1,t+4)=CS_tmp(iter1,t+4)*CL(MDLUP(iter1,4),t);
    end
end

for(o=1:n_o)
    for(d=1:n_d)
        if(o~=d)
            for(dt=1:n_dt)
                tt=P_MDLUP(1:P_MDLUP(PTR,I_MDLUP(o,d,dt)),I_MDLUP(o,d,dt));
                tmp1=CS_tmp(tt,4:10);
                [low_tmp1, col_tmp1]=size(tmp1);
                tmp2=zeros(low_tmp1,3);
                tmp2(:,1)=tmp1(:,1);
                tmp2(:,2)=tmp1(:,2)+tmp1(:,3)+tmp1(:,4)+tmp1(:,5)+tmp1(:,6)+tmp1(:,7);
                tmp3=zeros(low_tmp1,2);

```

```

for kkk=1:low_tmp1
    flag=0;
    if(kkk==1)
        tmp3(kkk,1:2)=tmp2(kkk,1:2);
        flag=1;
    elseif(kkk==low_tmp1)
        if(tmp2(kkk,3)~=1)
            tmp3(kkk,1:2)=tmp2(kkk,1:2);
            flag=0;
        end
    else
        if(tmp2(kkk,3)~=1)
            tmp3(kkk,1:2)=tmp2(kkk,1:2);
            flag=1;
        end
    end
    if(flag==1)
        for(ttt=kkk+1:low_tmp1)
            if(tmp3(kkk,1)==tmp2(ttt,1))
                tmp3(kkk,2)=tmp3(kkk,2)+tmp2(ttt,2);
                tmp2(ttt,3)=1;
            end
        end
    end
end

a=prod(tmp3(find(tmp3(:,2)~=0),2));
CS(o,d,dt)=a;
end
end
end
clear CS_tmp;

disp('Calculating CF')
for(o=1:n_o)
    for(d=1:n_d)
        for(dt=1:n_dt)
            if(PO(o,d,dt)==0)
                CF(o,d,dt)=0;
            else
                CF(o,d,dt)=CS(o,d,dt).^(1./PO(o,d,dt));
            end
        end
    end
end

disp('Calculating NewOD')
if(iter==1)
    NewOD(:,:,:)=OD(:,:,:).*CF(:,:,:);
    for(ii=1:7)
        for(jj=1:4)
            r=CO(ii,jj)/sum(NewOD(ii,:,jj))
            for(kk=1:50)

```

```

        NewOD(ii,kk,jj)=r*NewOD(ii,kk,jj);
    end
end
end
else
    NewOD(:,:,:)=NewOD(:,:,:).*CF(:,:,:);
    for(ii=1:7)
        for(jj=1:4)
            r=CO(ii,jj)/sum(NewOD(ii,:,jj));
            for(kk=1:50)
                NewOD(ii,kk,jj)=r*NewOD(ii,kk,jj);
            end
        end
    end
end
end

disp('Calculating MARE...')
tt=find(VA(:,1)~=0);
[a b]=size(tt);
EV1(iter,1)=(sum(abs(VA(tt,1)-VE(tt,1))./VA(tt,1))/a)*100;
EV1(iter,2)=(sum(abs(VA(tt,2)-VE(tt,2))./VA(tt,2))/a)*100;
EV1(iter,3)=(sum(abs(VA(tt,3)-VE(tt,3))./VA(tt,3))/a)*100;
EV1(iter,4)=(sum(abs(VA(tt,4)-VE(tt,4))./VA(tt,4))/a)*100;
EV1(iter,5)=(EV1(iter,1)+EV1(iter,2)+EV1(iter,3)+EV1(iter,4))/4

EV2(iter,1)=sum((VA(tt,1)-VE(tt,1)).^2);
EV2(iter,2)=sum((VA(tt,2)-VE(tt,2)).^2);
EV2(iter,3)=sum((VA(tt,3)-VE(tt,3)).^2);
EV2(iter,4)=sum((VA(tt,4)-VE(tt,4)).^2);
EV2(iter,5)=(EV2(iter,1)+EV2(iter,2)+EV2(iter,3)+EV2(iter,4))/4

clear VE_tmp;
save(Files(iter,:), 'NewOD', 'VE', 'VA', 'EV1', 'EV2');

end

toc
disp('The Program ends...');

```

Appendix D-2. MATLAB Code for GA-based Model

1. Main M-File

```
% gen_S3GA7.m
% This script demonstrates the use of the binary genetic algorithm

clc;clear;close all
format long;

global n_z; %the number of zones
global n_l; %the number of links
global n_dt; %the number of departure times
global n_t; %the number of time periods
global MDLUP;
global I_MDLUP;
global P_MDLUP;
global VA;
global MAX_f;

n_z=50; %the number of zones
n_l=3340; %the number of links
n_dt=4; %the number of departure times
n_t=8; %the number of time periods
MAX_f=1.0e+15;

disp('Data Loading...');
load vlb.mat;
load vub.mat;
load bits.mat;
load MDLUP.mat;
load I_MDLUP.mat;
load P_MDLUP.mat;
load VA.mat;

a=inputdlg('Mutation Probability?','Mutation Selection',1,{ '0.05'});
M=str2num(char(a));
clear a

a=inputdlg('Population size?','Population Selections',1,{ '200'});
P=str2num(char(a));
clear a

a=inputdlg('Generation Number?','Max. Generation Selections',1,{ '50'});
G=str2num(char(a));
clear a

a=inputdlg('Total OD?','Total OD Selections',1,{ '45000'});
TotalOD=str2num(char(a));
OutFile=char(a);
clear a;

MOE=1;%SSE
```

```

options = foptions(1);
options(11) = P; %Population size
options(12) = 0.5;% Default Pc
options(13) = M; %Mutation probability
options(14) = G; %Generation Number
options(2) = 0.00001; %Termination condition (no improvement in the Best)
options(3) = 0.00001; %Termination condition (no difference between the Best and mean)

[x,stats_sse,stats_mape,options,bf,lg] = genetic082('funn72',[],options,vlb,vub,bits>TotalOD,MOE);

clear MDLUP;
clear I_MDLUP;
clear P_MDLUP;
OutFileName = ['SSE' OutFile];
save(OutFileName);
disp("");
disp("");
disp('          !!!!!!! Matlab Running(GA) is over -- Thank you')

```

2. Function for GA Generation

```

function [xopt_pop,stats_sse,stats_mape,options,bestf,lgen] = genetic082(fun, ...
    x0,options,vlb,vub,bits,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10)

global MAX_f;
OPT_STOP = 0;

% Argument and error checking
if nargin<4,
    error('No population bounds given.')
elseif (size(vlb,1)~=1) | (size(vub,1)~=1),
    % Remark: this will change if algorithm accomodates matrix variables
    error('VLB and VUB must be row vectors')
elseif (size(vlb,2)~=size(vub,2)),
    error('VLB and VUB must have the same number of columns.')
elseif (size(vub,2)~=size(x0,2)) & (size(x0,1)>0),
    error('X0 must all have the same number of columns as VLB and VUB.')
elseif any(vlb>vub),
    error('Some lower bounds greater than upper bounds')
else
    x0_row = size(x0,1);
    for i=1:x0_row,
        if any(x0(x0_row,:)<vlb) | any(x0(x0_row,:)>vub),
            error('Some initial population not within bounds.')
        end % if initial pop not within bounds
    end % for initial pop
end % if nargin<4

if nargin<6,
    bits = [];

```



```

elseif (size(bits,1)~=1) | (size(bits,2)~=size(vlb,2)),
    % Remark: this will change if algorithm accomodates matrix variables
    error('BITS must have one row and length(VLB) columns')
elseif any(bits~=round(bits)) | any(bits<1),
    error('BITS must be a vector of integers >0')
end % if nargin<6

% Form string to call for function evaluation
if ~( any(fun<48) | any(fun>122) | any((fun>90) & (fun<97)) | ...
    any((fun>57) & (fun<65)) ),
    % Only alphanumeric implies must be a function
    evalstr = [fun 'x'];
    for i=1:nargin-6,
        evalstr = [evalstr,'P',int2str(i)];
    end
    evalstr = [evalstr, ''];
else
    evalstr = fun;
end

% Determine all options
% Remark: add another options index for type of termination criterion
if size(options,1)>1,
    error('OPTIONS must be a row vector')
else
    options = foptions(options);
    if options(11)==0,
        % Default size_pop
        options(11) = 1000;
    end
    if options(12)==0,
        % Default Pc
        options(12) = 0.5;
    end
    if options(14)==0,
        % Default max_gen
        options(14) = 50;
    end
end
PRINTING = options(1);
terminate = options(2);
terminate_mean = options(3);
size_pop = options(11);
Pc = options(12);
Pm = options(13);
max_gen = options(14);
% Ensure valid options: e.q. Pc,Pm,size_pop,max_gen>0, Pc,Pm<1
if any([Pc Pm size_pop max_gen]<0) | any([Pc Pm]>1),
    error('Some Pc,Pm,size_pop,max_gen<0 or Pc,Pm>1')
end

% Encode fitness function if necessary
ENCODED = any(any((([vlb; vub; x0]~=0) & ([vlb; vub; x0]~=1))) | ...
    ~isempty(bits);

```

```

if ENCODED,
    [fgen,lchrom] = encode_5(x0,vlb,vub,bits);
else
    fgen = x0;
    lchrom = size(vlb,2);
end
clear x0

% Display warning if odd number in initial population
if rem(size_pop,2)==1,
    disp('Warning: pop_size should be even. Adding 1 to population.')
    size_pop = size_pop + 1;
end

%disp('Form random initial population if not enough supplied by user');
if size(fgen,1)<size_pop,
    fgen = [fgen; int8(rand(size_pop-size(fgen,1),lchrom)<0.5)];% fgen <--- int8
end

xopt_pop = vlb;
bestf = Inf;
new_gen = fgen;% new_gen <--- int8
clear fgen;

%disp('Decoding the first generation');
if PRINTING>=1,
    if ENCODED,
        fp = decode_5(new_gen,vlb,vub,bits);
    end
end

disp('Start up main loop');
STOP_FLAG = 0;
for generation = 1:max_gen+1,

    options(10) = generation-1;

    disp('Evaluation...');
    % Decode first if necessary
    if ENCODED,
        x_pop = decode_5(new_gen,vlb,vub,bits);
    else
        x_pop = new_gen;
    end

    for i=1:size_pop,
        x = x_pop(i,:);
        if i==size_pop,
            OPT_STEP = 1;
        else
            OPT_STEP = 0;
        end
        [fitness1(i), fitness2(i)] = eval(evalstr);
    end
end

```

```

clear x;

[worst_fit,worst_index] = max(fitness1); %Finding the Worst
if (generation ~= 1) %Elitist Method
    new_gen(worst_index,:) = xopt_gen;
    fitness1(worst_index) = best_fit;
end
[best_fit,best_index] = min(fitness1); %Finding the Best

if(generation==1)
    if(P2==1)
        stats_sse = [best_fit max(fitness1) mean(fitness1) std(fitness1)]%Best, Worst, Mean, STD
        stats_mape = [min(fitness2) max(fitness2) mean(fitness2) std(fitness2)]%Best, Worst, Mean, STD
    else
        stats_mape = [best_fit max(fitness1) mean(fitness1) std(fitness1)]%Best, Worst, Mean, STD
        stats_sse = [min(fitness2) max(fitness2) mean(fitness2) std(fitness2)]%Best, Worst, Mean, STD
    end
else
    if(P2==1)
        stats_sse = [stats_sse; best_fit max(fitness1) mean(fitness1) std(fitness1)]
        stats_mape = [stats_mape; min(fitness2) max(fitness2) mean(fitness2) std(fitness2)]
    else
        stats_mape = [stats_mape; best_fit max(fitness1) mean(fitness1) std(fitness1)]
        stats_sse = [stats_sse; min(fitness2) max(fitness2) mean(fitness2) std(fitness2)]
    end
end

if best_fit < bestf,
    bestf = best_fit;
    xopt_pop = x_pop(best_index(1),:);
    xopt_gen = new_gen(best_index(1),:);
else
    % Remark: may want to regenerate to guarantee cost decrease
    % Remark: be careful not to get stuck in infinite loop
end

save ga_tmp;
clear x_pop;

% Display if necessary
% Remark: consider alternate printing options
disp('          Fitness statistics')
disp('Gen#      Best      Worst      Mean      Std. dev.')
if PRINTING>=1,
    if(P2==1)
        disp([sprintf('%5.0f %12.6g %12.6g ',generation-1, ...
            stats_sse(generation,1),stats_sse(generation,2)), ...
            sprintf('%12.6g %12.6g ',stats_sse(generation,3), ...
            stats_sse(generation,4))]);
    else
        disp([sprintf('%5.0f %12.6g %12.6g ',generation-1, ...
            stats_mape(generation,1),stats_mape(generation,2)), ...
            sprintf('%12.6g %12.6g ',stats_mape(generation,3), ...
            stats_mape(generation,4))]);
    end
end

```

```

    end
end

% Check for termination
% Remark: No improvement in the Best and No difference between the Best and mean
if terminate>0,
    if generation>10,
        if(P2==1)
            BEST = (stats_sse(generation-5,1)-stats_sse(generation,1))/stats_sse(generation-5,1)
            MEAN = (stats_sse(generation,3)-stats_sse(generation,1))/stats_sse(generation,1)
        else
            BEST = (stats_mape(generation-5,1)-stats_mape(generation,1))/stats_mape(generation-5,1)
            MEAN = (stats_mape(generation,3)-stats_mape(generation,1))/stats_mape(generation,1)
        end
        if ( (BEST < terminate) && (MEAN < terminate_mean) )
            STOP_FLAG = 1;
        end
    end
end
if STOP_FLAG | OPT_STOP,
    fprintf('\n')
    if STOP_FLAG,
        disp('Genetic algorithm converged.')
    else
        disp('Genetic algorithm terminated by user.')
    end
    return
end

disp('Reproduce...');
new_gen = normGeomSelect(new_gen,fitness1);

disp('Mate...');
new_gen = mate(new_gen);

disp('Crossover...');
new_gen = xover_yun(new_gen,Pc);

disp('Mutate...');
new_gen = mutate_5(new_gen,Pm);

if ENCODED,
    lgen = decode_5(new_gen,vlb,vub,bits);
else
    lgen = new_gen;
end

end % for max_gen

% Maximum number of generations reached without termination
if PRINTING>=1,
    fprintf('\n')
    disp('Maximum number of generations reached without termination')
    disp('criterion met. Either increase maximum generations')

```

```

    disp('or ease termination criterion.')
end

% end genetic

```

3. Function for Evaluation Function

```

function [f1,f2]=funn72(x,P1,P2)
%Name: funn72.m
%Evaluation function for Dynamic OD Estimation
%MOEs: SSE,MAPE
%P1->Total OD
%P2->MOE

global n_z; %the number of zones
global n_l; %the number of links
global n_dt; %the number of departure times
global n_t; %the number of time periods
global MDLUP;
global P_MDLUP;
global I_MDLUP;
global VA;
global MAX_f;
PTR=3000;

O1=1155;O2=1558;O3=1027;O4=3179;O5=3899;O6=4163;O7=4267;

% disp('Making OD to be Table format...');
SOD=zeros(n_z,1);
ptr=1;
for o=8:n_z
    SOD(o)=x(ptr);
    ptr=ptr+1;
end

TT=P1-O1-O2-O3-O4-O5-O6-O7;
SOD(:)=(SOD(:)./sum(SOD(:)))*TT;
SOD(1)=O1;SOD(2)=O2;SOD(3)=O3;SOD(4)=O4;SOD(5)=O5;SOD(6)=O6;SOD(7)=O7;

OD_Table=zeros(n_z,n_z);
for (o=1:n_z)
    for (d=1:n_z)
        if(o~=d)
            OD_Table(o,d)=x(ptr)/100;
            ptr=ptr+1;
        end
    end
    OD_Table(o,:)=OD_Table(o,:)/sum(OD_Table(o,:))*SOD(o);
end

OD=zeros(n_z,n_z,n_dt);
for (dt=1:3)
    for (o=1:n_z)

```

```

    for (d=1:n_z)
        if(o~=d)
            OD(o,d,dt)=OD_Table(o,d)*x(ptr)/100;
            ptr=ptr+1;
        end
    end
end
end

for (o=1:n_z)
    for (d=1:n_z)
        if(o~=d)
            OD(o,d,4)=OD_Table(o,d)-sum(OD(o,d,1:3));
        end
    end
end
clear x;

% disp('Calculating VE...');
VE=zeros(n_l,n_t);
for(o=1:n_z)
    for(d=1:n_z)
        if(o~=d)
            for(dt=1:n_dt)%
                tt=MDLUP(P_MDLUP(1:P_MDLUP(PTR,I_MDLUP(o,d,dt)),I_MDLUP(o,d,dt)),:);
                if(~isempty(tt))
                    a=size(tt,1);
                    for(iter_tt=1:a)
                        tt(iter_tt,5:12)=tt(iter_tt,5:12).*OD(o,d,dt);
                        VE(tt(iter_tt,4),1)=tt(iter_tt,5)+VE(tt(iter_tt,4),1);
                        VE(tt(iter_tt,4),2)=tt(iter_tt,6)+VE(tt(iter_tt,4),2);
                        VE(tt(iter_tt,4),3)=tt(iter_tt,7)+VE(tt(iter_tt,4),3);
                        VE(tt(iter_tt,4),4)=tt(iter_tt,8)+VE(tt(iter_tt,4),4);
                        VE(tt(iter_tt,4),5)=tt(iter_tt,9)+VE(tt(iter_tt,4),5);
                        VE(tt(iter_tt,4),6)=tt(iter_tt,10)+VE(tt(iter_tt,4),6);
                        VE(tt(iter_tt,4),7)=tt(iter_tt,11)+VE(tt(iter_tt,4),7);
                        VE(tt(iter_tt,4),8)=tt(iter_tt,12)+VE(tt(iter_tt,4),8);
                    end
                end
            end
        end
    end
end
end
end

for(k=1:n_l)
    for(t=1:4)
        VE(k,t)=0.973*VE(k,t+4)+VE(k,t);
    end
    for(t=5:6)
        VE(k,t)=VE(k,t)+0.827*VE(k,t-4);
    end
end
clear OD;

```

```

% disp('Calculating MOE...');
tt=find(VA(:,1)~=0);
N=size(tt,1);

if(P2==1)%SSE
    SSE=sum((VA(tt,1)-VE(tt,1)).^2);
    SSE=SSE+sum((VA(tt,2)-VE(tt,2)).^2);
    SSE=SSE+sum((VA(tt,3)-VE(tt,3)).^2);
    SSE=SSE+sum((VA(tt,4)-VE(tt,4)).^2);
    SSE=SSE/4;
    f1=SSE;

    MAPE=sum(abs(VA(tt,1)-VE(tt,1))./VA(tt,1))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,2)-VE(tt,2))./VA(tt,2))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,3)-VE(tt,3))./VA(tt,3))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,4)-VE(tt,4))./VA(tt,4))/N*100;
    MAPE=MAPE/4;
    f2=MAPE;
else%MAPE
    SSE=sum((VA(tt,1)-VE(tt,1)).^2);
    SSE=SSE+sum((VA(tt,2)-VE(tt,2)).^2);
    SSE=SSE+sum((VA(tt,3)-VE(tt,3)).^2);
    SSE=SSE+sum((VA(tt,4)-VE(tt,4)).^2);
    SSE=SSE/4;
    f2=SSE;

    MAPE=sum(abs(VA(tt,1)-VE(tt,1))./VA(tt,1))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,2)-VE(tt,2))./VA(tt,2))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,3)-VE(tt,3))./VA(tt,3))/N*100;
    MAPE=MAPE+sum(abs(VA(tt,4)-VE(tt,4))./VA(tt,4))/N*100;
    MAPE=MAPE/4;
    f1=MAPE;
end

```

4. Function for Normalized Geometric Selection

```

function[newPop] = normGeomSelect(oldPop,fitness)
% NormGeomSelect is a ranking selection function based on the normalized
% geometric distribution.

q=0.08; % Probability of selecting the best
e = size(oldPop,2); % Length of xZome, i.e. numvars+fit
n = size(oldPop,1); % Number of individuals in pop
newPop = zeros(n,e); % Allocate space for return pop
fit = zeros(n,1); % Allocates space for prob of select
x=zeros(n,2); % Sorted list of rank and id
x(:,1) =[1:1:n]'; % To know what element it was
fitness=fitness';

[y x(:,2)] = sort(fitness); % Get the index after a sort
r = q/(1-(1-q)^n); % Normalize the distribution, q prime
fit(x(:,2))=r*(1-q).^(x(:,1)-1); % Generates Prob of selection
fit = cumsum(fit); % Calculate the cumulative prob. func

```

```

rNums=sort(rand(n,1));           % Generate n sorted random numbers
fitIn=1; newIn=1;                % Initialize loop control
while newIn<=n                   % Get n new individuals
    if(rNums(newIn)<fit(fitIn))
        newPop(newIn,:)= oldPop(fitIn,:); % Select the fitIn individual
        newIn = newIn+1;                % Looking for next new individual
    else
        fitIn = fitIn + 1;              % Looking at next potential selection
    end
end
end

```

5. Function for Simple Crossover

```

function [new_gen,sites] = xover_yun(new_gen,Pc)
% XOVER Creates a NEW_GEN from OLD_GEN using crossover.
% [NEW_GEN,SITES] = XOVER(OLD_GEN,Pc) performs crossover
% procreation on pairs of OLD_GEN with probability Pc.
% Crossover SITES are chosen at random (re: there will be half as many SITES as there are individuals.

lchrom = size(new_gen,2);
n_col=size(new_gen,1)/2;

for(i=1:n_col)
    mask=rand(1,lchrom);
    for(j=1:lchrom)
        if(mask(1,j)<Pc)
            tmp=new_gen(2*i-1,j);
            new_gen(2*i-1,j)=new_gen(2*i,j);
            new_gen(2*i,j)=tmp;
        end
    end
end
end

```

6. Function for Simple Mutation

```

function [new_gen,mutated] = mutate_5(new_gen,Pm)
% MUTATE Changes a gene of the OLD_GEN with probability Pm.
% [NEW_GEN,MUTATED] = MUTATE(OLD_GEN,Pm) performs random mutation on the population
OLD_POP. Each gene of each individual of the population can mutate independently with probability Pm.
Genes are assumed possess boolean alleles. MUTATED contains the indices of the mutated genes.

[n_pop lchrom] = size(new_gen);

for(i=1:n_pop)
    mutated=rand(1,lchrom);
    for(j=1:lchrom)
        if(mutated(j)<Pm)
            new_gen(i,j) = int8(1-double(new_gen(i,j)));
        end
    end
end
end

```